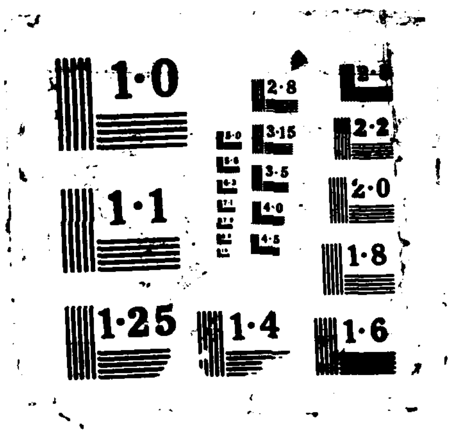
AD-A185 168 SIMPLIFICATIONS IN TEMPORAL PERSISTENCE: AN APPROACH TO THE INTRACTABLE D..(U) ILLINOIS UNIV AT URBANA COORDINATED SCIENCE LAB S A CHIEN SEP 87 1/1

UNCLASSIFIED UILU-ENG-87-2255 N00014-86-K-0309 F/G 12/9 NL


1·0
2·8
2·5
3·2
3·6
4·0
2·2
2·0
1·1
1·8
1·25
1·4
1·6

September 1987 UILU-ENG-87-2255

COORDINATED SCIENCE LABORATORY
College of Engineering

2

SIMPLIFICATIONS IN TEMPORAL PERSISTENCE: AN APPROACH TO THE INTRACTABLE DOMAIN THEORY PROBLEM IN EXPLANATION-BASED LEARNING

Steve Ankuo Chien

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

ADA185168

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS None	
2a. SECURITY CLASSIFICATION AUTHORITY N/A		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A			
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-87-2255		5. MONITORING ORGANIZATION REPORT NUMBER(S) N/A	
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois	6b. OFFICE SYMBOL *(If applicable)* N/A	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research	
6c. ADDRESS *(City, State, and ZIP Code)* 1101 W. Springfield Ave. Urbana, IL 61801		7b. ADDRESS *(City, State, and ZIP Code)* 800 N. Quincy St. Arlington, VA 22217	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Office of Naval Research	8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-86-K-0309	
8c. ADDRESS *(City, State, and ZIP Code)* 800 N. Quincy St. Arlington, VA 22217		10. SOURCE OF FUNDING NUMBERS	

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
N/A	N/A	N/A	N/A

11. TITLE *(Include Security Classification)*
SIMPLIFICATIONS IN TEMPORAL PERSISTENCE: AN APPROACH TO THE INTRACTABLE DOMAIN THEORY PROBLEM IN EXPLANATION-BASED LEARNING

12. PERSONAL AUTHOR(S)
Chien, Steve Ankuo

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Year, Month, Day)*	15. PAGE COUNT
Technical	FROM ____ TO ____	1987 SEPTEMBER	58

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB-GROUP	artificial intelligence, machine learning, explanation-based learning, failure-driven knowledge refinement, the intractabl domain theory problem, the frame problem, planning

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

In real-world domains, large amounts of knowledge are needed to adequately describe world behavior. With a complex domain theory, complete reasoning becomes a computationally intractable task. This is particularly true of knowledge-intensive learning techniques such as Explanation-Based Learning. This thesis describes an approach to problem-solving and learning designed to deal with computationally ill-behaved domains and a first implementation of this approach. In this approach, the system learns by constructing and generalizing causal explanations of observed plans. By using simplifications when necessary, the system avoids the intractability of complete reasoning. However, this introduces the possibility of learning imperfect plans. In order to deal with this contingency, the system monitors execution of these plans. When a discrepancy between the expected world state and the actual world state is detected, the system constructs an explanation for the discrepancy and uses this explanation to refine the faulty simplification. By using the real world to focus attention on incorrect simplifications, the system avoids the intractability of complete reasoning.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE *(Include Area Code)* / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

SIMPLIFICATIONS IN TEMPORAL PERSISTENCE: AN APPROACH TO THE INTRACTABLE DOMAIN THEORY PROBLEM IN EXPLANATION-BASED LEARNING

BY

STEVE ANKUO CHIEN

B.S., University of Illinois at Urbana-Champaign, 1985

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1987

Urbana, Illinois

SIMPLIFICATIONS IN TEMPORAL PERSISTENCE: AN APPROACH TO THE INTRACTABLE DOMAIN THEORY PROBLEM IN EXPLANATION-BASED LEARNING

Steve Ankuo Chien, M.S.
Department of Computer Science
University of Illinois at Urbana-Champaign, 1987
Gerald F. DeJong II, Advisor

In real-world domains, large amounts of knowledge are needed to adequately describe world behavior. With a complex domain theory, complete reasoning becomes a computationally intractable task. This is particularly true of knowledge-intensive learning techniques such as Explanation-Based Learning. This thesis describes an approach to problem-solving and learning designed to deal with computationally ill-behaved domains and a first implementation of this approach. In this approach, the system learns by constructing and generalizing causal explanations of observed plans. By using simplifications when necessary, the system avoids the intractability of complete reasoning. However, this introduces the possibility of learning imperfect plans. In order to deal with this contingency, the system monitors execution of these plans. When a discrepancy between the expected world state and the actual world state is detected, the system constructs an explanation for the discrepancy and uses this explanation to refine the faulty simplification. By using the real world to focus attention on incorrect simplifications, the system avoids the intractability of complete reasoning.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Gerald DeJong, whose encouragement, guidance, and patience aided this work immensely. In the CSL Explanation-Based Learning group he has created an excellent environment for learning and conducting research.

Thanks are also due to past and present members of the CSL AI research group, for innumerable discussions and suggestions. I would especially like to thank Ray Mooney for many worthwhile comments, particularly in the early stages of this work.

This work also benefitted from numerous discussions and diversions with Gregg Collins.

Comments on earlier drafts of this thesis by Brian Falkenhainer, Ray Mooney, and Dennis DeCoste greatly enhanced the readability of this document.

Last, but certainly not least, I would like to thank my family for their immeasurable love, support, and encouragement.

This research was conducted at the Coordinated Science Laboratory of the University of Illinois at Urbana-Champaign. A large portion of the implementation took part on equipment donated by IBM Corporation. Support was provided in part by a University of Illinois Cognitive Science/Artificial Intelligence Fellowship, an IBM Graduate Fellowship, and the Office of Naval Research under grant N-00014-86-K-0309.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

It is becoming increasingly apparent that large amounts of knowledge are essential for intelligent behavior in complex domains. This has led to a new trend in AI towards knowledge-intensive methods in many research areas such as expert systems, planning, natural language processing, high-level vision, and machine learning.

People perform robustly in complex domains; current AI systems cannot. The underlying problem is that in any complex domain, the amount of potentially relevant data is enormous. Even in game domains such as chess, the contrast between human and computer approaches is glaringly apparent. Computer chess programs evaluate potential moves by evaluating millions of possible board positions. Human chess experts evaluate potential moves by analyzing a few key potential avenues of play.

Yet how do experts rule out *irrelevant data? We believe* that the intelligent use of simplifications is critical to reasoning in real-world, complex domains. People frequently deal with complexity by using assumptions and abstractions. Consider planning to drive to work by a usual route. It is not difficult to concoct plausible sets of circumstances in which any stated plan would not succeed. For example, there might be construction on the route usually taken, there might be an accident, or there might be bad weather. Any of these occurrences would cause a slowdown in traffic, resulting in a late arrival at the office.

Without simplifications, designing even simple plans becomes intractable. Consider attempting to exhaustively prove that one of your plans would succeed or explicitly reasoning about every minute action needed to get you to your office if you drove. Worse yet, imagine trying to prove that all other agents would not interfere (knowingly or unknowingly) with your plan. If humans operated in this fashion they would spend unacceptably large amounts of time planning even the simplest of actions.

Nevertheless, most AI problem solvers have operated in exactly this fashion. Traditional AI planning systems [Sacerdoti77] produce plans by anticipating every possible interaction. Within this framework, problem-solving becomes a computationally demanding task for even simple problems (see [Chapman85] for an analysis of the computational complexity of planning).

A key aspect of effective use of simplifications in reasoning is the ability to recover from failures caused by inappropriate usage of simplifications. If one day major road construction causes your drive-to-work plan to result in a late arrival at work, you would likely try a different route the next day. This is because the failure would be likely to reoccur the next day. However, if a construction crew was merely painting lane stripes, they would probably finish before the next day; allowing the original plan to be used at that time. Any AI system that uses simplifications in reasoning must have the ability to: 1) detect situations in which simplifications are incorrect; and 2) recover appropriately in these cases.

This thesis describes an approach to dealing with complexity in problem-solving and learning. In this approach, problem-solving and learning are integrated through the selective use of simplifications. When initially learning plans, the system uses simplifying assumptions. During later usages of these plans, the system monitors execution to detect situations in which original assumptions are incorrect. When such a situation is detected, the system explains and corrects the faulty assumption. This capability allows the system to effectively use simplifications to deal with complexity in problem-solving and learning.

CHAPTER 2

USING SIMPLIFICATIONS IN PROBLEM-SOLVING AND LEARNING

This chapter outlines the approach to problem-solving and learning that has been developed. First, a brief overview of the schema-based problem-solving and explanation-based learning paradigms are given. Second, an approach to the use of simplifications in problem-solving and learning is discussed.

2.1 Schema-Based Problem-Solving

Problem-solving is a difficult task. Given an initial state, it involves finding and executing a sequence of actions to achieve a desired world state. Most AI planning systems have concentrated on building plans from base-level operators to achieve goals upon demand [Newell63, Sacerdoti77]. However, in a large plan space, as in real-world domains, the search involved in constructing plans from base-level operators makes this approach expensive.

A different approach to problem-solving involves reasoning from higher-level knowledge structures [Fikes72]. In this approach, a system solves problems by relying upon a library of high-level knowledge structures called schemata [Rumelhart80]. Schemata have preconditions, a goal, and an operator sequence. When the system is confronted with a problem-solving situation, it finds schemata which achieve the desired goal. These schemata are then tested to see if their preconditions are met. Any schemata which has its preconditions met can be used to achieve the goal by executing it's operator sequence.

However, the schema-based problem-solving paradigm requires a large number of these problem-solving schemata to achieve a high-level of performance. This presents a problem because it is often expensive or impossible for a programmmer to anticipate and hand-encode the large number of problem-solving schemata necessary for acceptable performance.

2.2 Explanation-Based Learning

One approach to acquiring problem-solving schemata is *Explanation-Based Learning* (EBL) [DeJong81, DeJong86, Mitchell86]. This approach has been applied to a wide range of domains including mathematical equation solving [Silver86], learning concepts in classical physics [Shavlik87a], robotics [Segre87a], mathematical theorem proving [O'Rorke87], integration problems [Mitchell83], learning functionality of artifacts [Kedar-Cabelli87], circuit design [Mitchell85], and narrative processing [Mooney85]. EBL differs from similarity-based methods such as [Langley81, Michalski83, Quinlan86] in that it involves learning new concepts through a knowledge-intensive analysis of causal dependencies in a training example.

Explanation-based learning techniques are particularly well adapted for use with schema-based problem-solving systems. In this approach, a system is given an initial domain knowledge consisting of lower-level knowledge structures such as operators and rules. The system then increases its performance by learning higher-level structures such as macro-operators by explaining and generalizing observations of its own or an external agent's problem-solving behavior.

By endowing a system with this capability, the cost or difficulty associated with hand-encoding a large number of schemata is reduced to encoding the base-level rules and operators needed to learn schemata. Because there are typically a large number of useful combinations of operators, the task of encoding base-level operators is more reasonable.

However, EBL suffers from the same difficulties as other knowledge-intensive AI techniques. In [Mitchell86], three classes of domain theory problems are discussed. First, *the incomplete domain theory problem* exists when the domain theory used by the learning system may not possess all of the information needed to properly explain observed events. With *the intractable domain theory problem,* a domain theory exists, but use of the theory to construct exhaustive proofs is computationally intractable. Last, the *inconsistent theory problem* exists when the domain theory can derive conflicting facts.

Because current EBL systems assume their domain theories are correct, any concepts they learn are also correct. This eliminates the need for refinement. Because these systems also assume tractable domains, they can use their domain theories exhaustively to learn complete concepts from a single input example.

2.3 Process Overview

This thesis describes a simplification–based approach to dealing with the intractable domain theory problem in explanation–based learning. In this approach, the system uses potentially incorrect simplifications in order to make the initial explanation process tractable. The system then relies upon refinement to detect and deal with cases in which the simplifications are incorrect.

This approach has two requirements. First, that a complete and sound domain theory exists. We presume that this domain theory exists at the level of rules and operators. This means that in theory the system can generate sound explanations given unbounded time and computational resources. Second, that a set of simplifications exists that can be used in portions of the the proof to make the process tractable[1] . Ideally these simplifications would be correct in almost all cases. This model of problem–solving and learning consists of five steps: *Initial Learning, Failure Detection, Failure Explanation, Failure Generalization and Packaging*, and *Knowledge Modification*.

Initial Learning: The system generates an approximate concept by first constructing an explanation of the goal concept, using the base (intractable) domain theory and simplifications when necessary. This explanation is a causal description of how the operators comprising the plan combine to achieve the goal. An important point is that the system must be able to later recognize and analyze the usage of simplifications in this explanation.

In complex domains, one of the major difficulties in reasoning is the so–called frame problem [McCarthy69]. This difficulty requires a system to update the world after various events have occurred; changing the facts that have changed and verifying the facts that have persisted. In our approach the system uses assumptions to reason about changes in the world efficiently.

The simplifications used in the explanation process (in the default logic notation of [Reiter80]) are defeasible inferences equivalent to that shown in Figure 2–1. IN(P,s) means that the fact P is believed by the system to be true in situation s. A fact P is said to contradict a fact Q if P is -Q or if P specifies that object x has value y for attribute z in a situation s and Q specifies

[1] Development of methods for learning simplifications or a general theory of simplifications is an area for future research. An initial taxonomy of simplifications is described in [Chien87a].

$$\forall\, P,si,sj,sk\ \ IN(P,si) \wedge \neg\exists\, Q\ [contradicts(Q,P) \wedge IN(Q,sj) \wedge precedes(si,sj) \wedge precedes(sj,sk)] \rightarrow IN(P,sk)$$

Figure 2-1: Defeasible Inference

that object x has value w for attribute z in situation s where y and w are ground values (not variables) and not equal. Precedes(a,b) means that the situation a temporally precedes the situation b.

One of the major concerns when making default inferences is conflicting defaults. Because the system uses default inferences in explanation, as opposed to prediction, powerful methods used to choose between conflicting defaults (such as minimality [McCarthy86] or strength-based models [Falkenhainer86, Haddawy86]) are not critical. However, the system does presume at least a weak method of dealing with conflicting defaults.[2]

The initial concept explanation is then generalized using the EGGS technique [Mooney86] and packaged for problem-solving in a manner that allows the simplifications used to be later analyzed if necessary. In order for the explanation to be used in problem-solving situations three portions of the explanation must be determined. First, the *plan preconditions* are the base-level supports of the explanation and are facts given to the system as being true in the initial situation. These facts represent the conditions under which the plan can be executed to achieve the goal. Second, the *operator sequence* is the sequence of operators executed to achieve the goal. These are the operators executed to use the plan. Finally, the *goal* is the desired world state achieved by execution of the plan.

Failure Detection: The use of simplifications in the initial learning process introduces the possibility of learning flawed plans. Consequently, the system needs a method of detecting and recovering from these cases. In our approach, the system relies upon goal failures to indicate situations in which learned concepts are incorrect. After execution of a plan, the system checks to

[2] The current implementation prefers the simplest explanation (least number of inferences plus assumptions).

see if the goal is achieved. If there is a failure to achieve a goal, then the plan used must be flawed and must be refined to prevent repetition of the failure.

Failure Explanation: After a goal failure is detected, the underlying causes for the failure are explained. This is done in two steps. First, the facts within the current situation that support the goal achievement are queried in the real world. These supports represent an easily verifiable level of failure backtracing. Any of these values which are not as expected are candidates as portions of the plan that require refinement. The system now attempts to explain the manner in which each of these values occurred. Because the domain theory is sound, any failure must be due to an invalid simplification. Each of the failed supports that can be explained will result in a new constraint upon the plan. Each of these explanations will be a specific proof of a fact Q that violates a simplification as described in Figure 2-1.

Failure Generalization and Packaging: In this step each explanation for a failed simplification is generalized using the EGGS technique [Mooney86]. Each failure explanation has a set of *failure preconditions*. These are the conditions beyond the preconditions for the plan which are the base-level supports for the failure. These represent the general conditions under which the designated failure will cause the plan to fail. This is a set of conditions under which the system previously believed that the plan would achieve the goal, but now realizes otherwise.

Knowledge Modification: Once the explanation and constraints for the failure have been computed, they are added to the preconditions of the original plan. The resultant plan is constrained by only being applicable in those cases where no set of failure preconditions are applicable. This is because each set of failure preconditions represents a case in which a simplification that the plan depends on is invalid. The revised set of plan preconditions for the plan is that the previous plan preconditions be met but none of the plan's failure precondition sets (including the newly created set) are met. The revised plan preconditions cause a situation in which the system incorrectly believed that the plan was applicable to become a situation in which the system correctly believes that a plan is not applicable. This means that an overly general plan has been specialized to become closer to the correct applicability description.

Additionally, by packaging failures in a manner independent from the plan with which they are learned, they may be used by other plans. This can occur in two ways. First, the failure explanation may also cause a failure in the other plan. If this can be detected before plan execution in a failure situation it can allow avoidance of the failure for the other plan. The second manner in which the failure structure can be used is to understand preventive measures. Given access to applicable failures, a plan can be checked to see if any of its actions prevent these failures. If so, these actions can be understood as preventing the appropriate failures.

CHAPTER 3

SYSTEM OVERVIEW

Our approach to using simplifications in problem-solving and learning has been tested by implementation of a prototype system. This system is implemented in Common LISP running on an IBM RT. This chapter and the rest of the thesis describe the simplification-based approach to dealing with complexity as embodied by the system.

3.1 System Architecture

The architecture of the learning system is shown in Figure 3-1. The system is initially given knowledge in the form of a complete and sound but intractable domain theory and simplifications. When the system is initially learning a plan, it receives an initial state, operator sequence, and goal as its input. Using the existing knowledge base of rules, facts, and operators, plus simplifications to make the understanding process tractable, the understander maintains the causal model which is the system's description of world events. The generalizer extracts the explanation for goal achievement from the causal model and generalizes the explanation to form a plan. This plan is then analyzed with respect to previously learned failures to check for preventive measures. The resultant structure is packaged into a plan and added to the plan library.

When the system is problem-solving, it receives a goal as its input. The planner selects an applicable plan from the plan library to achieve the goal. The executive then executes the plan, updating the causal model to reflect the changes in the world caused by plan execution. After the plan has been completed, the goal verifier verifies the achievement of the goal. If the plan fails to achieve the goal, the goal verifier backtraces to find the set of violated supports responsible for the failure. Because the domain theory is sound, these must be due to violated simplifications. The understander explains the reasons for these violated simplifications. The resulting explanations are then generalized and used to modify the original plan preconditions. If the plan achieves the goal, no refinement of plan preconditions is necessary.

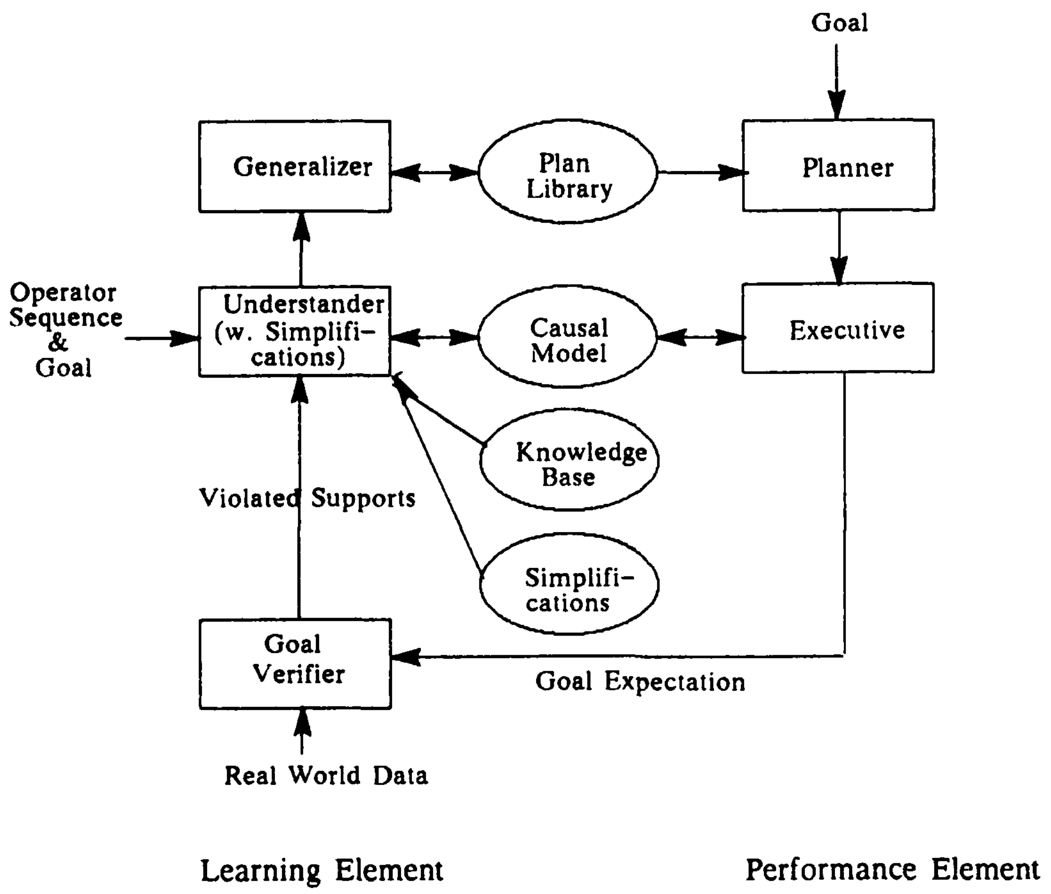


Figure 3-1: System Architecture

3.2 Sample System Performance

The implemented system has been tested on several examples from a manufacturing workshop domain. In this domain, the system has tools to drill, heat, roll, and cut various pieces. It is shown operator sequences to build simple mechanical assemblies and later constructs similar assemblies upon request.

A learning trace is shown below. In this example, the system is given the initial state, operator sequence, and goal. There are several pieces in the workspace. There is a plastic gear, a sheet, and a rod sitting on various workbenches. The goal is to build a widget, an assembly in which one end of a rod fits through a sheet and the other is attached to a gear as shown in Figure 3-2. Given the initial state and operator sequence, the system explains how the goal is achieved

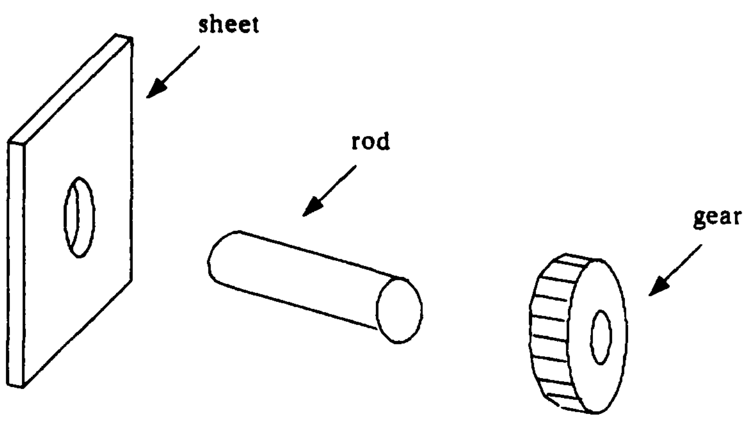


Figure 3-2: Widget Assembly

and learns a plan for building widgets. This explanation is constructed using several simplifying persistence assumptions. A trace of this episode is shown below.

```
Running: (RUN-EXAMPLE *W2*)
Initial State Consists of:
(AT R1 BENCH1)
(AT R2 BENCH1)
(AT G1 BENCH1)
(AT G2 BENCH1)
(AT P1 BENCH1)
(COMPOSITION R1 METAL)
(STATE R1 SOLID)
(STATE BASE SOLID)
(COMPOSITION G1 PLASTIC)
(SHAPE G1 GEAR)
(AT BASE BENCH1)
(SHAPE BASE SHEET)
Start state is state 0
operator 1 is (MOVE R1 OVEN)
operator 2 is (HEAT R1)
operator 3 is (MOVE R1 ROLLING-STATION)
operator 4 is (ROLL R1 10.0CM)
operator 5 is (COOL R1)
operator 6 is (MOVE R1 BENCH1)
operator 7 is (MOVE G1 DRILLING-STATION)
operator 8 is (DRILL G1 HOLE1 10.0CM)
operator 9 is (MOVE G1 BENCH1)
operator 10 is (MOVE BASE DRILLING-STATION)
operator 11 is (DRILL BASE HOLE2 10.1CM)
operator 12 is (MOVE BASE BENCH1)
operator 13 is (INSERT R1 G1 HOLE1)
operator 14 is (INSERT R1 BASE HOLE2)
Learning plan to achieve (WIDGET ?GEAR40645 ?PLANE37642 ?ROD41646 ?HOLE42647
?LOC38643 ?S139644).
Generalizing...
Packaging and Indexing...
Ordering Actions...
Marking assumption intervals...
Learned new plan WIDGET776
NIL
```

However, among the simplifications used to learn this plan is the assumption that the shape of the gear persisted from the start state to the final state. In reality, if the gear is plastic, this persistence depends upon the rod being cooled. Because of this simplification, the system does not deem the cooling step important to the goal achievement; resulting in the learning of a plan which omits the cooling step.

The system uses this plan to successfully construct several widgets. Then the system is asked to construct a widget with a plastic gear. The plan fails, because construction of a widget with a plastic gear depends upon the cooling step. The system explains this failure as the hot rod melting the gear, causing the deformation of the gear. The system then modifies the original plan so that it will not be used if the gear is plastic. This demonstrates the systems ability to recover from incorrect usage of simplifications. A trace of this precondition refinement example is shown below.

```
Running: (EXECUTE-EXAMPLE FAIL-EX)
Start state is state 0
Initial State is:
```

```
(AT R1 BENCH2)
(AT G1 BENCH3)
(AT BASE BENCH4)
(COMPOSITION R1 METAL)
(STATE R1 SOLID)
(SHAPE G1 GEAR)
(SHAPE BASE SHEET)
(COMPOSITION G1 PLASTIC)
(STATE G1 SOLID)
(STATE BASE SOLID)
Using plan WIDGET776 to achieve goal (WIDGET G1 BASE R1 HOLE3 BENCH3 ?S124752)
Operator 1 is (MOVE R1 OVEN)
Operator 2 is (HEAT R1)
Operator 3 is (MOVE R1 ROLLING-STATION)
Operator 4 is (ROLL R1 15.0CM)
Operator 5 is (MOVE R1 BENCH3)
Operator 6 is (MOVE G1 DRILLING-STATION)
Operator 7 is (DRILL G1 HOLE787 15.0CM)
Operator 8 is (MOVE G1 BENCH3)
Operator 9 is (MOVE BASE DRILLING-STATION)
Operator 10 is (DRILL BASE HOLE788 15.1CM)
Operator 11 is (MOVE BASE BENCH3)
Operator 12 is (INSERT R1 G1 HOLE787)
Operator 13 is (INSERT R1 BASE HOLE788)
Plan WIDGET776 completed.
Verifying achievement of functionality.
Goal does not verify.  Backtracing supports.
Verifying (AT BASE BENCH3) in situation 13
(AT BASE BENCH3) in situation 13 Verified in real world
Verifying (SHAPE G1 GEAR) in situation 13
(SHAPE G1 DEFORMED) is not expected value, will investigate.
Verifying (SHAPE BASE SHEET) in situation 13
(SHAPE BASE SHEET) in situation 13 Verified in real world
Verifying (THROUGH R1 BASE HOLE788) in situation 13
(THROUGH R1 BASE HOLE788) in situation 13 Verified in real world
Verifying (DIAMETER R1 15.0CM) in situation 13
(DIAMETER R1 15.0CM) in situation 13 Verified in real world
```

```
Verifying (HOLE HOLE788 BASE 15.1CM) in situation 13
(HOLE HOLE788 BASE 15.1CM) in situation 13 Verified in real world
Verifying (THROUGH R1 G1 HOLE787) in situation 13
(THROUGH R1 G1 HOLE787) in situation 13 Verified in real world
Verifying (DIAMETER R1 15.0CM) in situation 13
(DIAMETER R1 15.0CM) in situation 13 Verified in real world
Verifying (HOLE HOLE787 G1 15.0CM) in situation 13
(HOLE HOLE787 G1 15.0CM) in situation 13 Verified in real world
Attempting to explain (SHAPE G1 DEFORMED) in situation 13
Added censor SHAPE31232.
NIL
```

Although now the system will not fail by applying the plan without the cooling step to construct a widget with a plastic gear, it also cannot contruct a widget with a plastic gear. Next, the system is shown a training example with the cooling step (similar to the first example). Because the system now is aware of the melted gear failure, it justifies the cooling step as preventing this failure. Hence, it learns a new plan with the cooling step, demonstrating the systems ability to understand the relevance of previously observed failures to new plans.

CHAPTER 4

KNOWLEDGE REPRESENTATION

This chapter outlines the knowledge representation used by our approach. The representation for the system was designed with three goals in mind. First, the representation should be amenable to varying the amount of effort devoted to reasoning with a corresponding change in accuracy. Second, the representation should be sufficiently powerful so that the complexities of complete reasoning arise. Third, the representation (particularly for assumptions) should be explicit and clean enough to allow the system to analyze previously learned knowledge structures during refinement.

We use a representation roughly based upon situational calculus. In a given situation, a fact may be true, false, or unknown. Facts may refer to objects in the world and values, which are specifications of attributes for objects. There are two types of facts: definitional facts and situational facts. Definitional facts are true in all situations and specify properties or relations of values (e.g. the fact that 2 is greater than 1). Situational facts might be true only in a restricted set of situations and specify properties or relations among objects that may change over time.

A history consists of a sequence of situations connected by operators. In this representation, the assumption is made that all changes between temporally consecutive situations are instantaneous and brought about by the execution of a single operator. This means that within each situation the world is at a steady state.

Operators map from partial situations to partial situations. Operators are specified by an execution form, preconditions, effects, and create list. The execution form is the list of parameters that must be specified in order to execute the operator. The preconditions are a set of facts that must be true in the situation in which the operator is executed. Effects are facts that are true in the situation resulting from the execution of the operator. Note that because preconditions and effects can contain negations, they can also require the falsity of facts before or after execution of the operator. The create list is a set of objects created by the execution of the operator.

There are three types of rules in the system: *inter-situational rules*, *intra-situational rules*, and *persistence assumptions*. Inter-situational rules map from partial situations to partial situations. These rules consist of a consequent and an antecedent. The antecedent is a set of facts that must be true in a given situation in order for the rule to be applicable. The consequent is a fact that must be true in the state resulting from the execution of a certain operator in the antecedent state. These rules allow representation of operators whose effects depend upon state of the world in which the operator is executed (i.e. conditional effects). In intra-situational rules, the antecedent and consequent parts refer to facts in the same situation. Intra-situational rules allow the system to reason about properties such as transitivity, commutativity etc.

A persistence assumption is an potentially unsound inference that a fact persists from a previous situation. These are inferences of the form:[1]

$$\forall\ P,si,sj,sk\ \ IN(P,si) \wedge \neg\exists\ Q\ [contradicts(Q,P) \wedge\ IN(Q,sj) \wedge\ precedes(si,sj) \wedge\ precedes(sj,sk)]$$
$$\rightarrow IN(P,sk)$$

Formally, IN(P,s) means that the fact P is believed by the system in situation s. A fact P is said to contradict a fact Q if P is -Q or if P specifies that object x has value y for attribute z in a situation s and Q specifies that object x has value w for attribute z in situation s where y and w are ground values (not variables) and not equal. Precedes(a,b) means that the situation a temporally precedes the situation b. Intuitively, this rule states that a fact P can be assumed to persist to a later situation provided it is not explicitly contradicted by a belief in an intervening situation.

Definitional facts, operators and rules are the initial knowledge given to the system. Plans represent goal-directed problem-solving knowledge learned by the system. Censors represent corrections to plans produced by analyzing observed failures and are similar to the censors described in [Winston86]. By processing examples consisting of a complete initial situation specification, goal, and operator sequence, the system is able to contruct the higher-level knowledge structures of plans and censors.

[1] Currently these inferences are performed by specialized LISP code. Using a single general inference engine for both defeasible and standard inferences and a declarative representation for defeasible inferences are areas for future work.

An explanation for a fact is an instantiation of a set of applicable rules and assumptions which supports the given fact. A rule in an explanation is a valid support for a fact if the instantiation of the consequent is the fact and for each fact in the rule antecedent there is a fact in the causal model that directly supports it or there is an applicable supporting rule or assumption. An assumption in an explanation supports a fact if the most recent value for the fact is the same as the assumed value.

Due to this recursive definition, every explanation support must be grounded in an initial state fact or the effect of an operator. This operator in turn will eventually be grounded in an initial state fact.

A plan is a representation for the manner in which a constrained sequence of operators achieves a desired world state. A plan consists of a *goal state, plan preconditions*, an *operator sequence*, a *causal explanation*, and a set of *censors*. The *goal state* is a partial situation that the system believes will be true after the execution of the plan. The *plan preconditions* are a partial world specification that is required for the proper execution of the plan. The *operator sequence* is a list of the operators which cause the goal state to be true, ordered as in the input example. The *causal explanation* is a description of how the plan preconditions allow the operator sequence to achieve the goal state.

A censor is a subsequent correction to a flawed plan. A censor consists of an *explanation*, an *expected value*, an *observed value*, and a set of *failure preconditions*. The *explanation* is a causal structure describing a case in which a plan is incorrect. In this case, the plan explanation states that a certain fact has a designated value. However, due to a faulty persistence assumption, there is a different value in the real world. There are three portions of this explanation which are important. First, the *expected value* is the fact value which is supported by the plan explanation. Second, the *observed value* is the value which is supported by the censor explanation. Finally, the *failure preconditions* are a set of facts which must be true in order for the censor explanation to be applicable.

CHAPTER 5

EXPLANATION CONSTRUCTION AND GENERALIZATION

This chapter details the initial learning phase of the simplification–based learning process. First, the causal model maintained by the system is described. Next the support network generalization is discussed. Finally, the packaging of plans is described.

5.1 Causal Model Construction

The causal model is a history representing the systems perception of the state of the world at various points in time and a causal description of how the execution of operators alters the world state. This causal model maintains support links for the systems belief of facts in the manner of truth–maintenance systems [Doyle80]. The system learns general plans from specific initial state descriptions and operator sequences through analysis of the dependencies maintained by the causal model. The initial state, operator sequence, and goal used to construct the causal model might be given explicitly to the system as training examples, or could be derived from observation of a domain expert solving problems.

Understanding facts means that the system knows why the state represented by the fact is true. This can be done in two ways, as an initial condition, or as an effect of an action (possibly via inference rules). When the system receives an input fact it attempts to explain why the fact is true in the world. If the system succeeds in explaining the occurrence of the fact, the fact and supporting explanation for the occurrence of the fact are added to the causal model. If no explanation can be found, the fact is added to the causal model marked as being an input fact.

For operators, understanding means knowing why each of the preconditions is true, and knowing the effects of the operator. As operators are input to the system, they are added to the causal model by first incorporating the preconditions and then adding the effects. The preconditions are incorporated into the causal model in a manner identical to processing of input facts. The effects of the operator are added to the causal model by being asserted true in the situation representing the world state resulting from and caused by the execution of the operator

After the initial state description and operator sequence, the system is given the goal that is achieved by the example plan. The system then attempts to construct an explanation for how the plan steps achieve the goal (as with any other input fact). If the system can construct an explanation, it learns a plan from the observed operator sequence.

5.2 An Example

In order to clarify the explanation construction and generalization process, a small plan for rolling a bar will now be presented. In this example, the following initial state, goal, and observed operator sequence are input to the system:

Initial State (s0):

1)	(at bar21 bench4)
2)	(composition bar21 metal)
3)	(state bar21 solid)
4)	(at sheet22 bench1)
5)	(shape sheet22 sheet)

Goal State:
(diameter bar21 10.0cm)

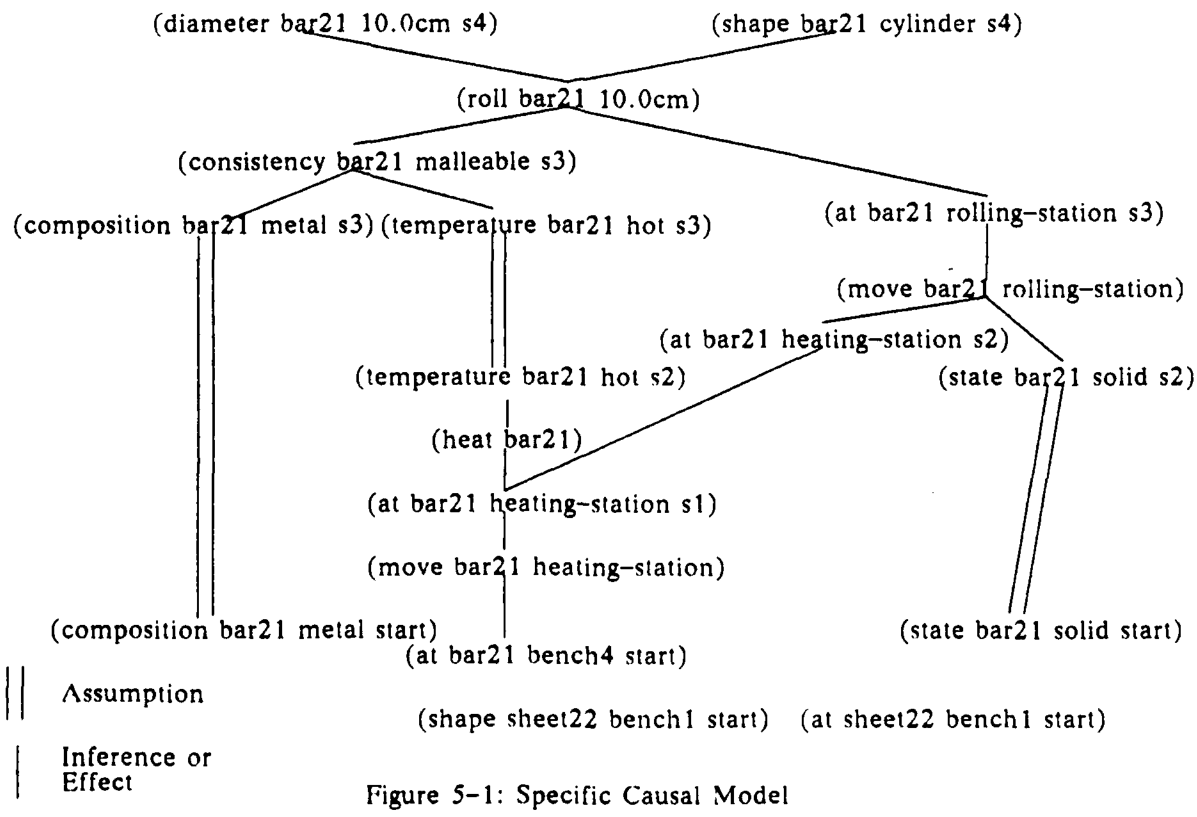
Operator Sequence:

1)	(move bar21 heating-station)	move the bar to the heating station
2)	(heat bar21)	heat the bar
3)	(move bar21 rolling-station)	move the bar to the rolling-station
4)	(roll bar21 10.0cm)	roll the bar to a diameter of 10.0cm[1]

Upon receiving the first input action, the system verifies that the preconditions of the move operator are satisfied by noting that the previous location of the bar was bench4 and that the consistency of the bar was solid in the start state. It then computes the effect of the move and asserts the fact (at bar21 heating-station) in s1, the state caused by executing the first operator in s0. It then processes the second operator. The one precondition of the heat operator is that the object to be heated be at the heating station. Since this fact was achieved by operator 1, the

[1] Because the system's domain model does not represent that rolling an object does not change its volume, the system does not know that the bar must have initially had a diameter of 10cm or that the length of the bar will change.

system confirms this easily. The system then asserts the effect of the heating operation, that the temperature of bar21 is hot in situation 2. Next the system processes the third operator. It verifies that the state of bar21 is solid by assuming that this fact persists from situation 0 to situation 2. It then adds the effect of the move – that the bar is at the rolling-station in situation 3. Next the system processes the fourth operator – the roll operation. The roll operator has two preconditions. First, that the object being rolled is at the rolling-station. This is confirmed easily as it was asserted by the move operator 3. The second precondition is that the object have malleable consistency. This is verified by a rule which says that if an object is hot and metal, it is malleable. In order to apply this rule, the system assumes that the hot temperature of the bar persists from state 2 and that the metal composition of the bar persists from state 0. Having verified the preconditions of the operator, the system now asserts the two effects of the roll: 1) the bar now has a cylindrical shape; and 2) the diameter of the bar is now 10.0cm. The causal model constructed for this example is shown in Figure 5-1.



(diameter bar21 10.0cm s4)
(shape bar21 cylinder s4)
(roll bar21 10.0cm)
(consistency bar21 malleable s3)
(composition bar21 metal s3)
(temperature bar21 hot s3)
(at bar21 rolling-station s3)
(move bar21 rolling-station)
(at bar21 heating-station s2)
(temperature bar21 hot s2)
(state bar21 solid s2)
(heat bar21)
(at bar21 heating-station s1)
(move bar21 heating-station)
(composition bar21 metal start)
(state bar21 solid start)
(at bar21 bench4 start)
Assumption
(shape sheet22 bench1 start)
(at sheet22 bench1 start)
Inference or Effect

Figure 5-1: Specific Causal Model

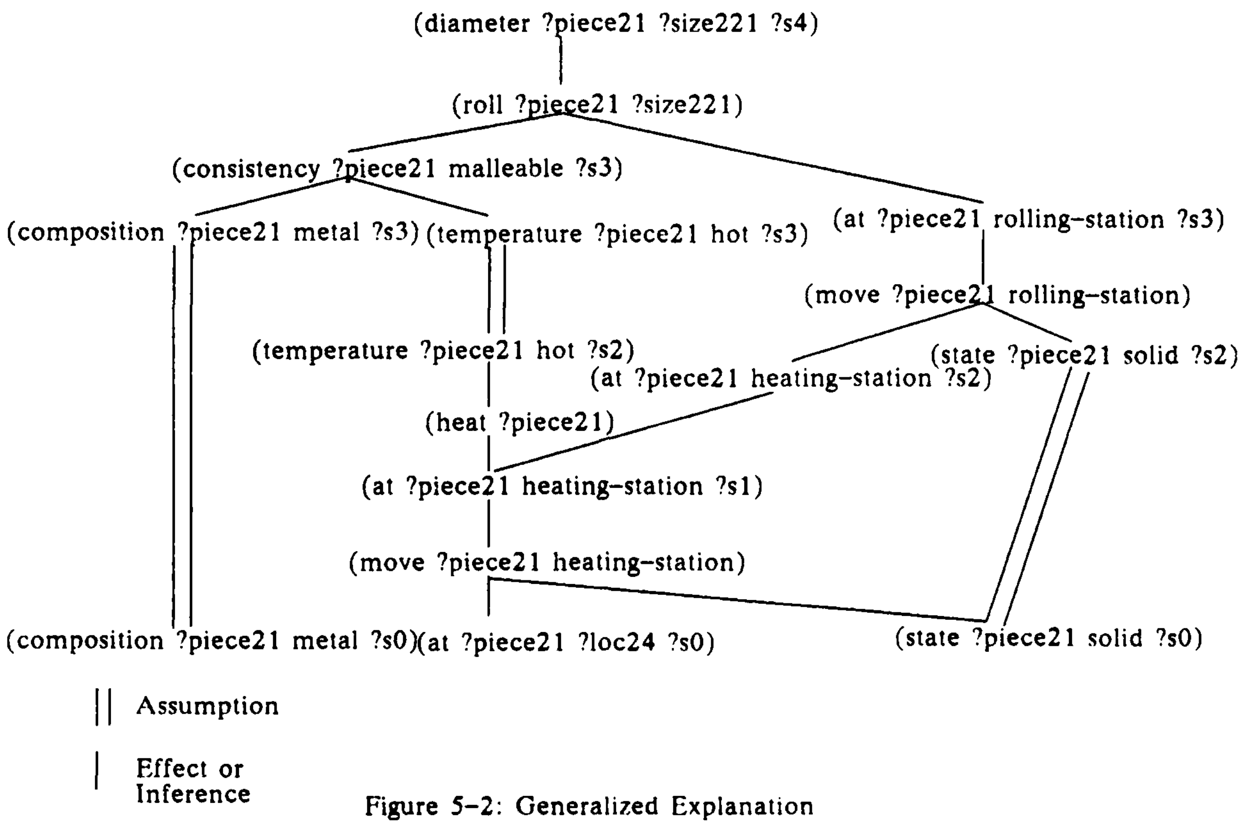
In the example, the system is told that the goal is that the diameter of the bar be 10.0cm. Because the goal is already supported by the causal model, no additional explanation is necessary. If the goal fact were not supported, the system would attempt to construct an explanation for why the goal specification was achieved before proceeding with the generalization phase.

5.3 Explanation Generalization

After the system has explained goal achievement, a plan can be learned from the training example. The generalization and packaging phase consists of three steps. First, the bindings for the general plan are computed. This is done using an adaptation of the EGGS technique [Mooney86]. Next, the plan is analyzed with respect to previous plans (discussed in Section 6.2). Finally, the new plan is packaged and indexed to enable usage in problem-solving situations.

The first step in the plan learning process is the computation of general bindings. This is done by stepping through the causal network, tracing the actions and inferences supporting the goal. At each support, the assertion that is supported must match the assertion caused by the support. For example, the fact that an action supports must be equal to the appropriate effect of the action. This constraint is enforced by the bindings given by unifying these two forms. Persistence assumption supports are handled by unifying the assertions but not the time tags. This is because an assertion can be supported by a previously believed assertion through a persistence assumption. The generalized causal structure produced by this process represents the most general case in which this explanation will be valid. The general explanation for the rolling-bar example is shown in Figure 5-2.

The final step in learning a plan is packaging. In this section the procedure necessary to learn a plan in isolation is described. The process of analyzing related plans will be discussed in Section 6.2. There are three parts of the general causal explanation which are needed to use the plan in a problem-solving manner: the *goal*, *operator sequence*, and *plan preconditions*. The *goal* is the general form supported by the causal explanation. This describes the partial world situation that is achieved by the plan. When this world state is desired, the plan may be useful. The *operator sequence* is the general list of operators in the observed operator sequence that appear in the general causal explanation. The operator sequence represents the operators that must be executed in order to execute the plan. The *preconditions* are the initial state facts grounding the general



(diameter ?piece21 ?size221 ?s4)
(roll ?piece21 ?size221)
(consistency ?piece21 malleable ?s3)
(composition ?piece21 metal ?s3)
(temperature ?piece21 hot ?s3)
(at ?piece21 rolling-station ?s3)
(move ?piece21 rolling-station)
(temperature ?piece21 hot ?s2)
(state ?piece21 solid ?s2)
(at ?piece21 heating-station ?s2)
(heat ?piece21)
(at ?piece21 heating-station ?s1)
(move ?piece21 heating-station)
(composition ?piece21 metal ?s0)
(at ?piece21 ?loc24 ?s0)
(state ?piece21 solid ?s0)
Assumption
Effect or Inference

Figure 5–2: Generalized Explanation

causal explanation. These facts represent the partial world situation in which the plan can be executed to achieve the goal.

Figure 5–3 shows the goal, operator sequence, and plan preconditions for the rolling-bar plan. Note that many extraneous details of the training example have been removed, such as: the desired diameter of the piece can be any value, there need not be a sheet shaped object at bench4, and the bar used need not be bar21.

GOAL:	(diameter ?piece21 ?size221)
OPERATOR SEQUENCE:	(move ?piece21 heating-station) (heat ?piece21) (move ?piece21 rolling-station) (roll ?piece21 ?size221)
PLAN PRECONDITIONS:	(composition ?piece21 metal) (at ?piece21 ?loc24) (state ?piece21 solid)

Figure 5-3: Plan Goal, Operator Sequence, and Plan Preconditions

CHAPTER 6

REFINEMENT

This chapter describes the refinement capabilities of the system and consists of two main sections. The first section describes the manner in which the system refines plan preconditions to recover from incorrect usage of simplifications in the initial learning process. The second section describes the manner in which the system can understand measures to prevent previously observed failures.

6.1 Recovering from Incorrect Simplifications in Plans

The usage of simplifications in initial learning introduces the possibility of learning flawed plans. In order to deal with this contingency, the system must have a method of detecting and correcting such flaws. In our approach, the system relies upon goal failures to indicate situations in which plans require refinement. In this case, the system incorrectly believes that a plan will succeed in a given situation; but due to a faulty simplification the plan fails. Subsequently, the system explains and generalizes the case in which the plan will fail, and restricts the original plan preconditions so as to prohibit selection of the plan in these situations. The resultant refinement specializes an overly general plan by making a case in which the system incorrectly believed that the plan would apply become a case in which the system correctly believes that the plan will not apply.

This section begins with an overview of the problem-solving and failure detection processes used by the system. Next, the plan precondition refinement process is described. Finally, an example is shown to illustrate this process.

6.1.1 Problem-Solving

In problem-solving mode, the system uses previously learned plans to solve problems. The problem-solving component of the system receives a goal as its input. Its task is to select and execute a plan to achieve this goal. There are two constraints that an applicable plan must satisfy. First, the plan must achieve the desired goal. This constraint is satisfied by retrieving the set of plans which are indexed as achieving the goal state. Second, it must be possible to execute the plan; and execution of the plan should achieve the desired world state. The plan preconditions of a plan are the set of initial facts that support the explanation for goal achievement (from the initial learning process) plus the conditions necessary to avoid previously observed failures (from censors). The plan preconditions must be met in order for a plan to be applicable in a problem-solving situation.

If no applicable plan is found, the system cannot achieve the goal. The system does not modify plans to achieve goals similar to indexed goals. The system also does not actively plan to achieve unsatisfied plan preconditions or prevent anticipated failures This is because attempting to modify plans is computationally expensive and requires the system to have significant problem-solving capability. Instead, the system waits until observing an applicable solution and then learns the desired plan.

Once the system has chosen an applicable plan, the system executes the plan. In this process, the system outputs the instantiated operators to the world simulator. Concurrently, the causal model is updated to reflect the effects of the operators.

After completing the plan, goal achievement is verified Currently this is done by an ad hoc module outside of the system. If the goal is achieved, the plan is successful, and no refinement is necessary. However, if the goal is not achieved, the system has failed and must refine the plan to block repetition of the failure.

6.1.2 Refining Plan Preconditions

After a goal failure has indicated a flawed plan, the system modifies the original plan preconditions in a three step process:

Verification of Current State Supports: The system checks the supports for the goal in the current situation. These are the set of facts which should be true in the current world state supporting the goal achievement. However, some of these facts are not true - thus causing the failure.

Explanation of Support Violation: The system forms a causal explanation for why the support states are violated.

Censor Packaging: The system packages this information into a censor. This censor is a knowledge structure to be added to the plan to prevent future inappropriate selection of the plan.

After a failure has been detected, the system backtraces to find the lowest-level supports for goal achievement in the current situation. This is done by tracing backwards in the causal network until any type of support other than an intra-situation rule is found. The system then queries the world simulator to verify these values (the system currently assumes either direct verifiability for these values or the existence of an outside module to indirectly verify these values). This process yields the set of violated supports. Note that because we do not allow defeasible inferences within the same situation, if the goal is not achieved in the current situation, one of the current situation supports must be invalid.

Given the set of violated assertions, the system then finds the real world value of each assertion and attempts to construct an explanation for how this value occurred. This is done in a backward-chaining manner using inter-situation rules, intra-situation rules, and persistence assumptions. The resultant explanation(s) are then generalized using the EGGS algorithm.

Each generalized explanation is then packaged into a censor. First, the conditions leading to each failed assumption are computed by tracing the appropriate explanation back to the leaves. These leaf justifications may be effects of actions in the plan or they may be initial situation facts. Facts suppported by action effects are met whenever the plan is executed and hence are not stored. However, the correspondences between objects and values in the plan and censor are computed by unifying these forms and become the bindings field of the censor. Initial situation facts are collected and become the failure preconditions field of the censor. These are the conditions under which the plan was previously thought to be applicable; but an analysis of the failure has shown otherwise. Next, the general unexpected value that the censor explanation supports becomes the censor observed value and the violated expected value of the plan becomes the expected value. The resultant censor represents a case in which a persistence assumptions in the original plan can be violated by circumstances possibly in conjunction with actions executed in the plan.

6.1.3 An Example

In order to clarify the precondition refinement process, an example is described. In this example, the system has learned a general widget plan. In this plan, the system begins with a metal rod, a plastic gear, and a sheet. It first heats and rolls the rod. Next, it drills holes into the gear and sheet. It then inserts the rod into the gear for a tight friction fit and through the sheet with a loose fit that allows the rod to spin. The goal, operator sequence, and plan preconditions for this plan are shown in Figure 6-1.

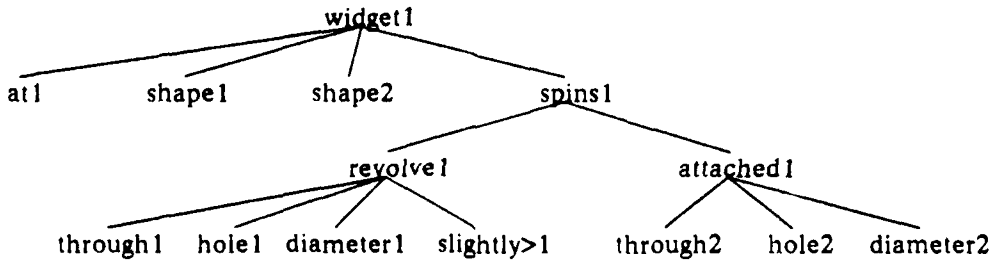
```
GOAL:         (widget ?x32546 ?obj4571 ?obj7608 ?hole5572 ?loc44547 ?s124752)

OPERATOR         (move ?obj7608 oven ?situation440)
SEQUENCE:        (heat ?obj7608 ?situation441)
                 (move ?obj7608 rolling-station ?situation453)
                 (roll ?obj7608 ?size6535 ?situation464)
                 (move ?obj7608 ?loc44547 ?situation506)
                 (move ?x43546 drilling-station ?situation519)
                 (drill ?x43546 ?hole5534 ?size6535 ?situation520)
                 (move ?x43546 ?loc44547 ?situation544)
                 (move ?obj4571 drilling-station ?situation557)
                 (drill ?obj4571 ?hole5572 ?size6573 ?situation558)
                 (move ?obj4571 ?loc44547 ?situation582)
                 (insert ?obj7608 ?x43546 ?hole5534 ?situation606)
                 (insert ?obj7608 ?x4571 ?hole5572 ?situation633)

PLAN              (at ?x43546 ?old-loc45524 ?gen-sit782)
PRECONDITIONS:    (slightly> ?size6573 ?size6535)
                  (composition ?obj7608 metal ?gen-sit781)
                  (at ?obj7608 ?old-loc45444 ?situation440)
                  (state ?obj7608 solid ?situation440)
                  (shape ?obj4571 sheet ?gen-sit780)
                  (shape ?x43546 gear ?gen-sit779)
                  (at ?obj4571 ?old-loc45561 ?gen-sit778)
                  (state ?obj4571 solid ?gen-sit777)
```

Figure 6-1: Plan Goal, Operator Sequence, and Plan Preconditions

The system attempts to use this plan in the initial situation shown in Figure 6-2. The system attempts to build a widget and fails. As a result, the system backtraces the current situation supports for the widget plan used. The current situation portion of the widget plan is shown in Figure 6-3. The system verifies each of the leaves in this support network. It then determines that all of the current-situation supports are met except that g1 is not gear shaped. It determines that in the real world the shape of the g1 is deformed in situation s13.

```
(at r1 bench2)
(at g1 bench3)
(at base bench4)
(composition r1 metal)
(state r1 solid)
(shape g1 gear)
(shape base sheet)
(composition g1 plastic)
(state g1 solid)
(state base solid)
```

Figure 6-2: Example Initial State



widget1	(widget g1 base r1 hole3 bench3 s13)
shape1	(shape g1 gear s13)
at1	(at base bench3 s13)
shape2	(shape base sheet s13)
spins1	(spins g1 base r1 hole788 s13)
revolve1	(revolve r1 base hole788 s13)
attached1	(attached r1 g1 s13)
through1	(through r1 base hole3 s13)
hole1	(hole hole788 base 15.1cm s13)
diameter1	(diameter r1 15.0cm s13)
slightly>1	(slightly> 15.1cm 15.0cm)
through2	(through r1 g1 hole787 s13)
hole2	(hole hole787 g1 15.0cm s13)
diameter2	(diameter r1 15.0cm s13)

Figure 6-3: Goal Supports within Current Situation

Next, the system attempts to explain why the gear is deformed in s13 in the current example. The system derives the explanation shown in Figure 6-4. The deformed shape of the gear is

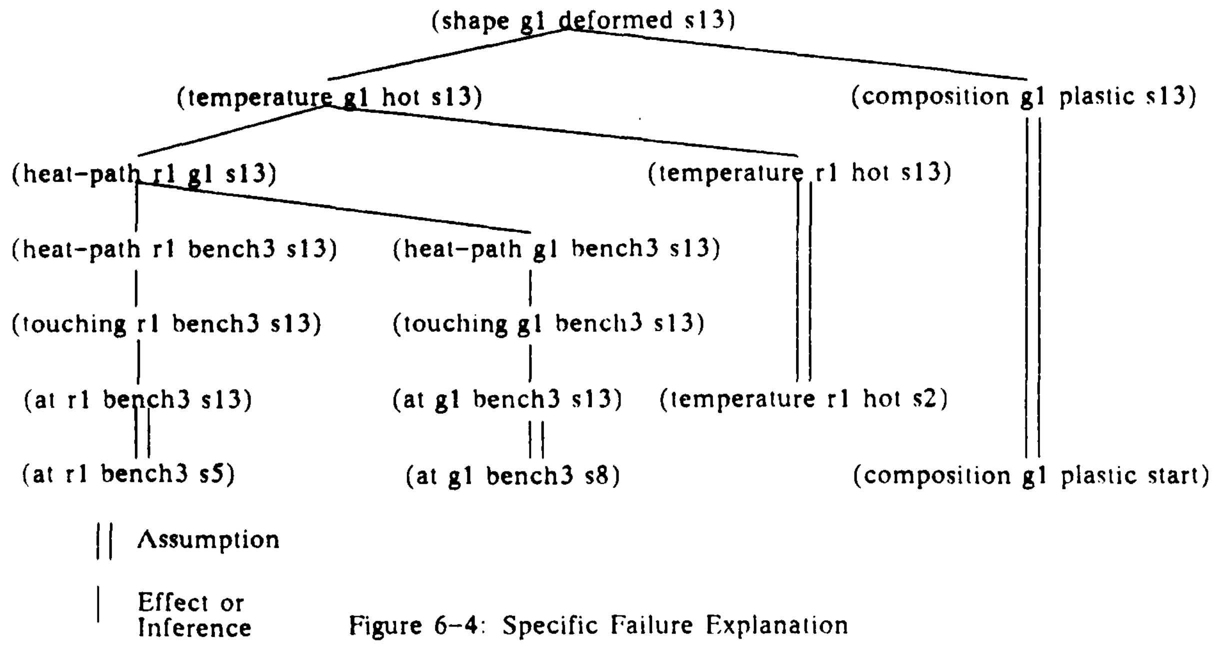
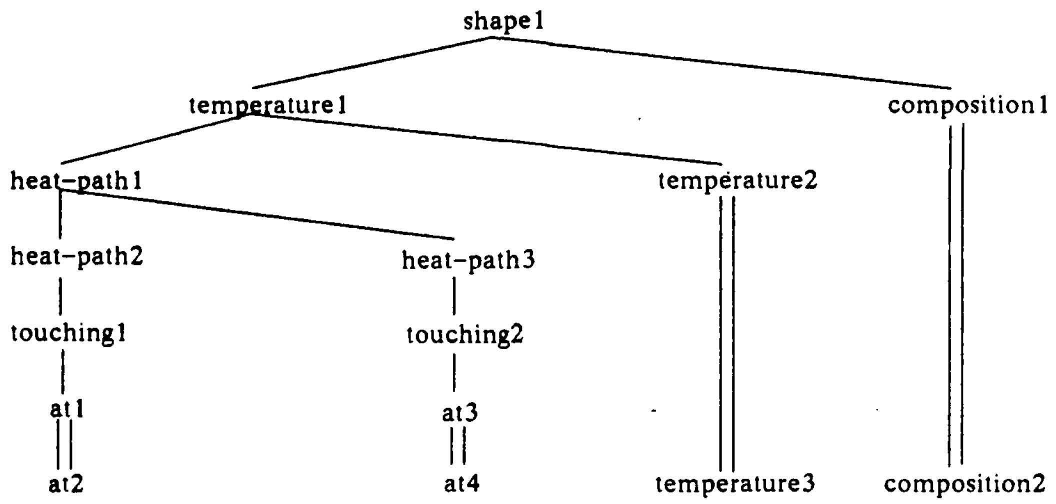


Figure 6-4: Specific Failure Explanation

explained as follows. The gear g1 is plastic and became hot in situation s13 and thus the shape of g1 was deformed. G1 is plastic in s13 because it was plastic in the start situation and this property persisted through situation s13. G1 became hot because there was a heat-path from the metal rod r1 to g1 in s13 and the r1 was hot in s13. R1 was hot in s13 because it was hot in s2 as an effect from the heating action and this state persisted through situation s13. There was a heat-path from g1 to r1 in s13 because there was a heat-path from r1 to bench3 in s13 and also a heat-path from g1 to bench3 in s13. There was a heat-path from r1 to bench3 in s13 because r1 was touching bench3 in s13. R1 was touching bench3 because r1 was at bench3 in s13, which persisted from s5. Likewise, there was a heat-path from g1 to bench3 in s13 due to g1 touching bench3 in s13, which was supported by g1 being at bench3 in s13, which persisted from g1 being at bench3 at s8.

The resulting failure explanation is then generalized using the EGGS algorithm. This is done in exactly the same manner as plan generalization, except that no actions appear within the failure explanation. The general form of the explanation is shown in Figure 6-5. This general explanation of the failure will form the explanation of the resulting censor.



|| Assumption

| Effect or Inference

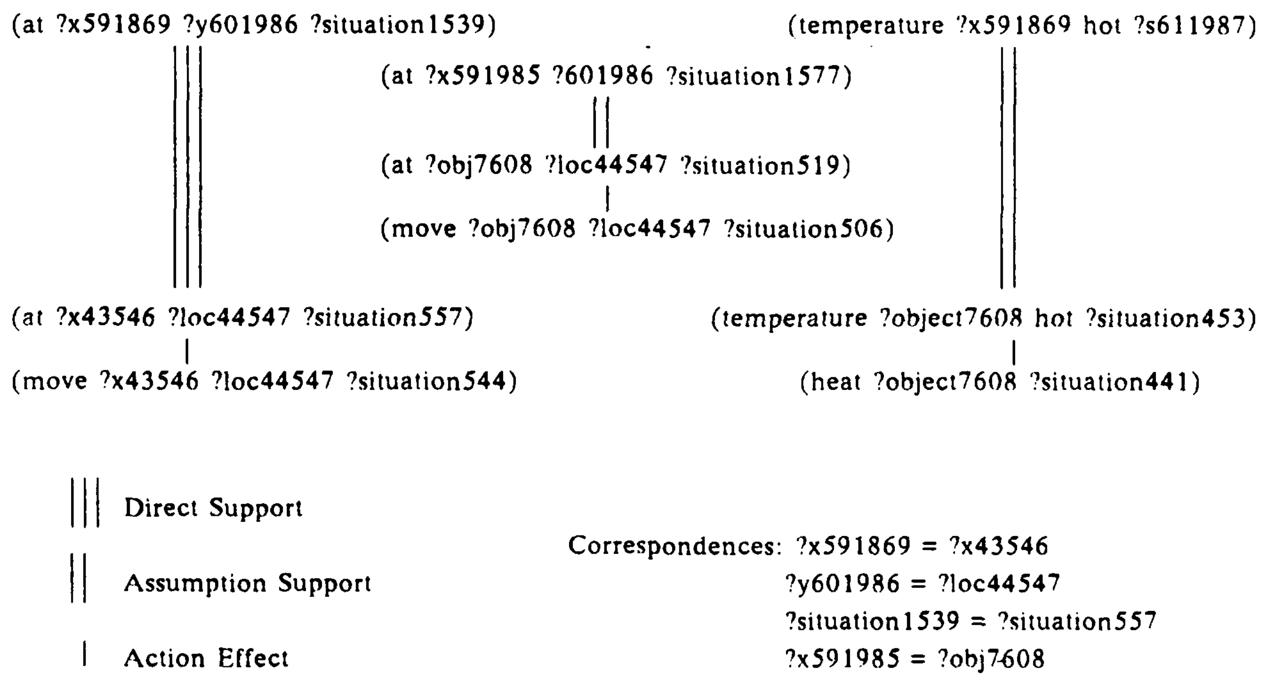
shape1	(shape ?x591985 deformed ?s611987)
temperature1	(temperature ?x591985 hot ?s611987)
composition1	(composition ?x591985 plastic ?s611987)
heat-path1	(heat-path ?x591869 ?x591985 ?s611987)
heat-path2	(heat-path ?x591869 ?y601986 ?s611987)
heat-path3	(heat-path ?x591985 ?y601986 ?s611987)
touching1	(touching ?x591869 ?y601986 ?s611987)
touching2	(touching ?x591985 ?y601986 ?s611987)
at1	(at ?x591869 ?y601986 ?s611987)
at2	(at ?x591869 ?y601986 ?situation1539)
at3	(at ?x591985 ?y601986 ?s611987)
at4	(at ?x591985 ?y601986 ?situation1577)
temperature2	(temperature ?x591869 hot ?s611987)
temperature3	(temperature ?x591869 hot ?situation1485)
composition2	(composition ?x591985 plastic start)

Figure 6-5: General Failure Explanation

Next, the failure preconditions are found by examining the leaf nodes of the general failure explanation. Of the leaves in the failure explanation, all but the fact that the gear is plastic are effects of actions in the plan, and hence will be true in any plan execution. The fact that the rod is at the same location as the gear is a result of the rod being moved to location ?y601986 (the fifth action in the plan) and the gear being moved to location ?y601986 (the eighth action in the plan). The fact that the rod is hot is an effect of the heating step (the second action in the plan). Because the fact that the gear is plastic is a start condition it becomes the failure preconditions of the censor.

After the failure preconditions are found, the correspondences between the failure and plan are computed. This involves unifying the leaves of the failure explanation which are supported by action effects with the general effects of the actual actions in the plan. These correspondences (a set of variable bindings) become the binding field of the new censor.

In the melted gear failure, the action supports are shown in Figure 6-6. Of these, the gear


(at ?x591869 ?y601986 ?situation1539)
(temperature ?x591869 hot ?s611987)
(at ?x591985 ?601986 ?situation1577)
(at ?obj7608 ?loc44547 ?situation519)
(move ?obj7608 ?loc44547 ?situation506)
(at ?x43546 ?loc44547 ?situation557)
(temperature ?object7608 hot ?situation453)
(move ?x43546 ?loc44547 ?situation544)
(heat ?object7608 ?situation441)
Direct Support
Assumption Support
Action Effect

Correspondences: ?x591869 = ?x43546
?y601986 = ?loc44547
?situation1539 = ?situation557
?x591985 = ?obj7608

Figure 6-6: Correspondences between Plan and Failure Explanation

being moved to the assembly location is a direct support. Hence the correspondences from this support are given by unifying the location effect with the location node in the failure explanation. The other two supports are assumption supports. The correspondences for these supports are computed by unifying all but the situation portion of the facts. The resulting correspondences are also shown in Figure 6-6.

Next, the observed value for the censor is determined by finding the general form that the failure exp`ination supports. The expected value is retrieved by examining the general form in the assumption from the original plan that the censor blocks. A censor name is then created from the attribute name for the observed value. Finally, the censor is created with the computed values and added to the relevant plan. The failure preconditions, expected value, and observed value of the censor are shown in Figure 6-7.

Censor Values

Failure Preconditions: ((composition ?x591985 plastic start))

Expected Value: (shape ?x591985 gear ?s1241784)

Observed Value: (shape ?x591985 deformed ?s611987)

Figure 6-7: Failure Preconditions, Expected Value, and Observed Value of Censor

The censor affects plan applicability as follows. In problem-solving situations in which the censor applies, the plan will not be selected due to the expectation of plan failure represented by the censor. In this manner, the original plan has been modified to reflect the knowledge learned by analyzing the failure.

6.2 Analyzing Plans with Respect to Previous Failures

Consider planning a surprise party for Bill while not allowing Bill to find out about the party. Or getting to your office without your boss seeing you. Or carrying four glasses over to the table without spilling the water in them. All of these tasks involve prevention. More specifically, prevention is executing actions to achieve a partial world specification while not allowing another partial world specification from becoming true.

This section describes the process by which the system understands how previously observed failures relate to new plans. In our approach, the system understands preventive measures by analyzing examples in which an observed plan succeeds in a case where the system expects the plan to fail. While it would be desirable to understand preventive measures before having observed the circumstances which they prevent, postulating and checking potential failures is a combinatorially explosive process and would require exhaustive reasoning. In order to avoid this difficulty, the system waits until it has a known instance of a plan which prevents the failure. The system is then faced with a contradiction. The censor predicts that the plan will fail and yet the system has observed the plan succeeding. Given this situation, the system can then focus upon the portions of the specific failure explanation that can be defeated and examine the plan for operators that successfully prevent these conditions.

After a plan is generalized and packaged as discussed in Chapter 5, the system compares the newly learned plan to known censors. If the system determines that the current plan could fail in the manner described by a censor, it analyzes the current plan to see if it contains operators to prevent the failure. If so, these operators are marked as preventing the failure. If the failure is not prevented, the system modifies the plan preconditions of the new plan to prohibit its usage in situations where the failure will occur.

6.2.1 Understanding Relevant Failures

This analysis of relevant failures consists of three steps:

Determine Relevant Censors: The system determines the set of censor whose action supports are supported by actions in the current plan.

Check for Prevent Operators: The system analyzes the current plan to determine if it contains operators that prevent the designated failure.

Modify the New Plan: If the failure is prevented, mark the appropriate operators are doing so. If the failure is not prevented, modify the plan preconditions as to not allow the plan to be used in cases where the failure is applicable.

The first step in understanding failures relevant to a new plan is to find potentially relevant censors. This is done by finding the set of all censors that are supported by actions which occur in the current plan which is computed by matching (unifying) the action supports of censors of previously learned plans to actions in the plan currently being learned. If it is possible to match, in order, all of the actions that occur in the censor to actions in the observed plan, then the system checks if the censor observed value will violate an assumption in the current plan. If this is the case, then the censor is relevant to the current plan.

The second step involves checking for preventive actions in the current plan. If a censor has its conditions apply in the current example, yet the plan did not fail, it is desirable to understand how the failure was avoided. The failure explanation consists of sound inference rules and defeasible persistence assumptions. If the failure conditions and actions are met, and the failure does not occur, one of the persistence assumptions in the failure explanation must be invalid. The system examines the actions in the operator sequence to determine which persistence assumption is defeated. This blocking of the failure is then explained and the new plan is marked as avoiding the failure in the manner detailed by the blocking explanation. The explanation for the blocking of the failure is now part of the plan and these additional constraints become part of the plan.

The third step involves modifying the plan with respect to the applicable censors. If the failure represented by the censor is not prevented by the plan, the censor is added to the censors for the newly learned plan. This censor will prevent the plan from being used in cases where the censor applies. This possibility is shown in Figure 6-8. In this circumstance, the plan contains an

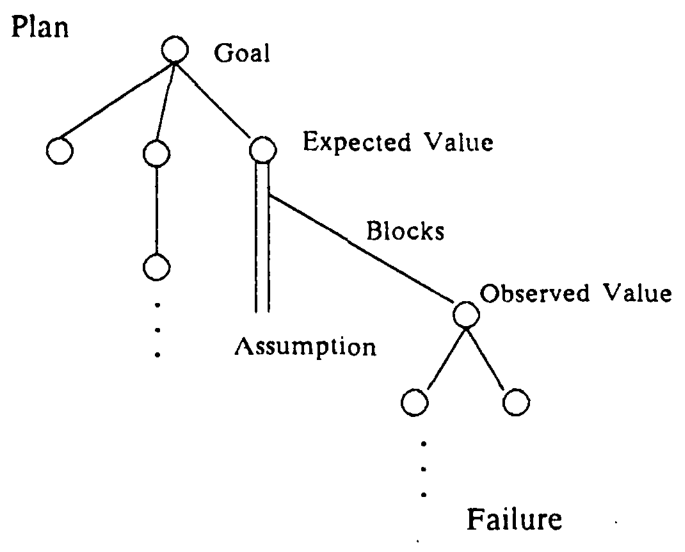


Figure 6-8: Failure Blocks Assumption in Plan

assumption that a fact will persist over some interval. However, under some set of circumstances, an explanation holds (the failure explanation in Figure 6-8) which details how this fact does not persist. As a result, the plan will fail in these circumstances.

However, the failure may have been prevented by operators in the plan. In this case, the failure contains an assumption which is blocked by an action in the plan. This possibility is shown in Figure 6-9. In this case there is an assumption in the failure explanation that a fact persisted

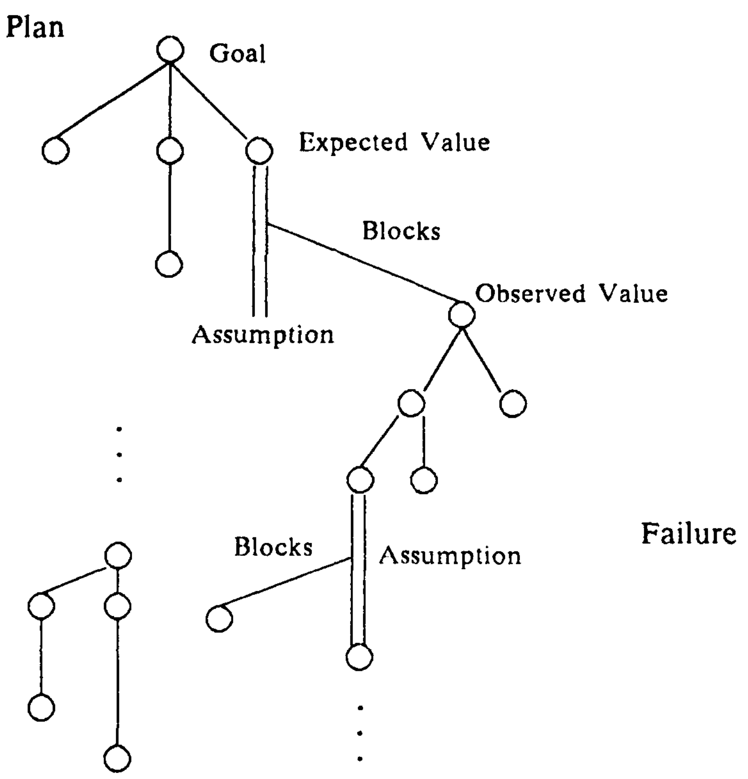


Figure 6-9: Plan Blocks Assumption in Failure

over some interval. However an effect of an operator in the plan (possible through inferences) blocks the persistence of this fact. Hence the failure explanation is invalid.

6.2.2 An Example

In order to clarify the process of incorporating previously observed failures, the example from Section 6.3 is extended. In this example, the system has learned a plan to assemble several pieces into a widget. One of the pieces, a metal bar, is heated and rolled to fit into a hole drilled into a gear. This plan, as detailed in Section 6.1.3, contains a censor representing a failure in which the gear is plastic and is deformed by being placed at the same location as the heated rod.

The system is now shown an initial state and operator sequence in which the metal rod is cooled after being rolled to the correct size. This initial state and operator sequence is shown in Figure 6-10. When the cool metal rod is inserted into a plastic gear, no deformation occurs. An

```
Initial State:      (at r21 bench2)
                    (at r2 bwnch1)
                    (at g312 bench3)
                    (at g2 bench1)
                    (at p1 bench1)
                    (composition r21 metal)
                    (state r21 solid)
                    (state base16 solid)
                    (composition g312 plastic)
                    (shape g312 gear)
                    (at base16 bench1)
                    (shape base16 sheet)

Operator Sequence:
                    (move r21 oven)
                    (heat r21)
                    (move r21 rolling-station)
                    (roll r21 10.0cm)
                    (cool r21)
                    (move r21 bench1)
                    (move g312 drilling-station)
                    (drill g312 hole1 10.0cm)
                    (move g312 bench1)
                    (move base16 drilling-station)
                    (drill base16 hole2 10.1cm)
                    (move base16 bench1)
                    (insert r21 g312 hole1)
                    (insert r21 base16 hole2)
```

Figure 6-10: Intial State and Operator Sequence for the Cooling Plan Example

analysis reveals that the deformed gear failure from the previous plan applies to the current plan and the system deduces that the failure conditions were met. Hence, a persistence assumption in the failure must have been blocked by the new plan. The system notes that the cooling step blocks the persistence assumption about the heat of the rod, thus preventing the deformed gear failure. This circumstance is shown in Figure 6-11.

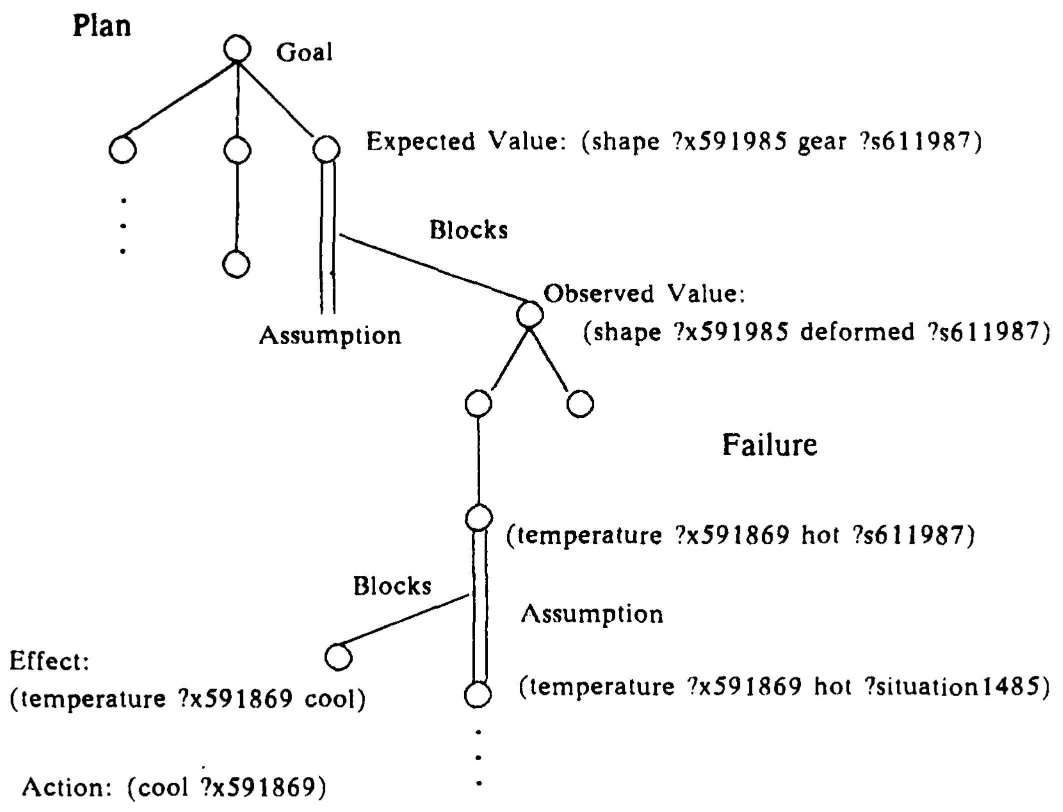

Plan
Goal
Expected Value: (shape ?x591985 gear ?s611987)
Blocks
Observed Value:
(shape ?x591985 deformed ?s611987)
Assumption
Failure
(temperature ?x591869 hot ?s611987)
Blocks
Assumption
Effect:
(temperature ?x591869 cool)
(temperature ?x591869 hot ?situation1485)
Action: (cool ?x591869)

Figure 6–11: Cooling Action in Plan Blocking Failure

Because the cooling step did not appear in the general explanation for goal achievement, it was previously thought by the system to be an unimportant action. While in theory the system had the base-level knowledge necessary to understand this prevention upon initially learning the plan, this would have required the system to postulate the possibility of the melted gear failure which is a computationally expensive process. However, after the system has observed the failure and learned the corresponding censor, the censor predicts that the plan will fail and observed data is that the plan has succeeded. This contradiction serves to focus the system's attention on portions of the failure that can be blocked; allowing tractable understanding of the preventive measure.

The next step is to find the appropriate constraints on the plan required to prevent the failure. This is done by propagating the constraints between the expected and observed values in the failure explanation (yielding the correspondences between the objects and values in the failure and the objects and values in the plan), computing the constraint that the effect of the cool action must block the persistence assumption in the failure explanation, and adding the supporting causal explanation for the cooling step to the plan (i.e. the the supporting causal model for the preconditions of the cooling step.)

CHAPTER 7

COMPARISONS TO OTHER WORK

There are several similar approaches to problem-solving and learning. A number of incremental knowledge-based learning systems have been constructed. In addition to the ARIES system (the predecessor of the current system), there has been work on incremental knowledge-based learning approaches by Bennett, Hammond, Doyle, and Gupta.

7.1 Comparison to the ARIES System

The system described in this thesis evolved from the ARIES system [Chien86, Chien87b]. The ARIES (for Automated Refinement and Indexing of Explanatory Schemata) system is an incremental EBL system designed to extend the GENESIS system which understands narratives. The ARIES system refines schemata for understanding narratives describing criminal plans. Because of the computational intractability of exhaustively proving that other agents do not interfere with the planning agents plans, the ARIES system assumes that other agents do not not perform actions to interfere with the original agent's plans (i.e. counter-plan). When the system processes a narrative describing a situation in which counter-planning causes a plan failure, it refines the initial (flawed) plan by explaining and generalizing the observed counter-plan. The newly created counter-plan explanation can be used to understand narratives describing similar failures due to counter-planning and to understand measures taken to prevent similar failures.

In order to illustrate the refinement process used by the ARIES system and its relation to the algorithm used by the current system, an example processed by the ARIES system will now be described. The ARIES system is given a schema describing a kidnapping plan learned by the GENESIS system. This schema assumes that other agents will not interfere with the kidnappers actions. Next, the ARIES system processes the conceptual representation for a narrative describing how a kidnapper is arrested and imprisoned after revealing his identity to the captive. The ARIES system notes that the kidnap plan has failed because the kidnapper's preserve-freedom goal is violated. Consequently it explains and generalizes the manner in which the captive learned the identity of the kidnapper and how this led to the arrest and emprisonment of the kidnapper.

This explanation structure is added to the kidnapping schema and can now be used to understand similar failures and measures taken to prevent similar failures.

The similarities between the approach used by the ARIES system and the current system should be quite clear. Both system use initial schema representations with simplifications. The ARIES system assumes no counter-planning while the present system makes persistence assumptions. When confronted with a goal failure, both systems explain and generalize how the goal failure occurred. With the ARIES system, processing the narrative describing the kidnapper being jailed causes it to refine the flawed kidnapping schema. The present system notes that it has failed to produce a working widget, causing it to refine the widget plan to explain how the melted gear caused a failure.

Although the basic approach used by the ARIES system is the same as in the present system, the present system differs from the ARIES system in three important ways. First, the ARIES system refined schemata for understanding. The current system refines schemata used in problem-solving. Second, the current system has a more explicit representation and cleaner semantics for assumptions. Third, the ARIES system assumed no counter-planning as a simplification whereas the the current system uses persistence assumptions as simplifications.

7.2 Comparison to Other Work

One type of simplification is numerical approximation. In this area, Bennett [Bennett87] uses a set of qualitative domain rules as an approximation to a complex numerical relationship, allowing learning in intractable mathematical domains. In some cases the learned approximation can be verified by substituting back into the original quantitative formula.

Bennett's approach differs from the approach described in this thesis in two ways. Bennett's work deals with numerical simplifications, this work addresses conceptual simplifications. No clear conceptual abstraction is the analogue of order of magnitude or qualitative approximation. Second, while Bennett's approximation allows for validation of approximations, there is no mechanism for repairing flawed approximations.

Hammond's CHEF system [Hammond86] also addresses the problem of refining existing flawed plans. The CHEF system plans for new problems by retrieving similar plans and modifying them to produce initial plans. When one of these modified plans fails, the CHEF system constructs an explanation for why the plan failed. CHEF then uses this explanation to classify the conditions that lead to the failure under a set of *planning TOPs*. Planning TOPs are general classifications of failures that suggest classes of actions to correct the failure. CHEF then repairs the plan by using a suggested fix. CHEF also creates a rule using the generalized conditions that cause the failure in order to anticipate problems in similar future situations. When subsequent plans involving the same features that caused the failure are processed, the modified plan will be retrieved due to the fact that it solves the anticipated problem.

There are several important differences between our approach and that used by the CHEF system. First, CHEF presumes a powerful problem-solver that is able to solve anticipated problems. Our approach makes no such requirement, and hence must learn plan repairs from observation. Second, CHEF uses a static classification of planning failures to repair plans. Our approach does not require this type of knowledge. Third, because CHEF uses an episodic memory, it can determine with absolute certainty whether a plan episode failed or succeeded. In a schema-based system, certain instantiations of a plan may succeed, while others will fail. Finally, because the CHEF system corrects plans that it generates (vs. understanding an external agents behavior), CHEF has access to information from the plan generation phase (i.e. why certain operators were chosen, etc.) whereas in our approach this information must be built up by an understanding module.

Another area of related work is Doyle's work on learning causal descriptions of devices [Doyle86]. In this approach, the system has several levels of detail in its domain theory. The system uses this domain theory to explain the behavior of causal mechanisms. At the more detailed levels of the domain theory, the theory is more accurate. As predictions made by the current mechanism description are contradicted by observations made by the system, the system moves to a more detailed level of description.

Doyle's approach differs from ours in two important ways. First, Doyle's system uses schematic descriptions to explain the behavior of mechanisms whereas in our approach a system builds explanations from a more basic domain theory of rules and operators. This means that Doyle's approach depends upon a predefined abstraction hierarchy, while our approach requires no such organization. Furthermore, Doyle's system learns causal descriptions of mechanisms whereas we are concerned mainly with understanding and refining plans.

Gupta's recent work on explanation-based failure recovery [Gupta87] addresses using failures to refine over-generalized operator application rules in planning. In this approach the system explains failures and determines which subgoals the system was attempting to achieve while causing the failure. His system then adds the constraint of avoiding the failure to this subgoal. When his system replans for the subgoal, it will avoid the failure due to the newly learned constraint.

There are several important differences between our approach and Gupta's. First, because Gupta system learns from its own problem-solving traces it has access to information on choices made in the plan generation process. Our approach deals with inferring the purposes of operators within the plan. Second, Gupta's approach deals with failures that restrict choice of operators to achieve goals. It does not address addition of operators to a plan to prevent a failure (as in the addition of the cooling operator to prevent the melted plastic gear failure). Additionally, it is not clear that posting constraints at choice points extends to the problem of failures involving multiple dependent choices of operators.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

This project has been a useful vehicle for investigating problem-solving and learning in complex domains. It has demonstrated the feasibility of using simplifications to learn in computationally ill-behaved domains. The representation and methods used are general enough such that the approach can be extended to other examples or domains. However, there are a number of interesting research areas that remain for future work:

(1) First, a taxonomy of simplifications should be developed. Initial work towards this goal is described in [Chien87a]. A general system could then be constructed that would use meta-level reasoning to select and apply simplifications to problem-solving in a wide range of application areas. Critical to this goal is the development of a uniform representation of simplifications.

(2) A related area for work is extending the simplification-based approach to other types of defeasible inferences. While the implemented system uses only persistence assumptions, it seems reasonable that other types of default rules could also be used.

(3) Currently the system is given a set of simplifications with a domain theory. An interesting area for research would be development of techniques to allow a system to learn simplifications for use in reasoning.

(4) Another related area for work is efficient execution monitoring and error detection. In certain cases, a system may be able to isolate a failure to a set of assumptions, but not be able to further pinpoint the cause of the failure. An open research area is to develop a strategies to efficiently monitor these suspect assumptions during future problem-solving episodes. Alternatively, a system could perform some form of testing or experimentation to further investigate the nature of the flaw (such as [Rajamoney85, Rajamoney87]).

(5) Generalizing order of actions in plans is yet another area for research [Mooney87]. Currently, the system executes plans with the same ordering of action that was observed.

However, in an environment with limited resources or parallel execution, a system should be able to re-order actions to more efficiently execute plans. For example, several steps may be able to be executed in parallel. Or alternatively, certain machines may only be available at certain times. Efficiently dealing with either of these circumstances would require an understanding of the true constraints on order of actions in plans.

(6) Representing plans at multiple levels of abstraction is an additional area of research. Choosing the appropriate level of generality at which to represent plans is not an easy task [Keller87, Segre87b, Shavlik87b]. More specific plans have the advantage of being easier to execute; whereas more general plans are applicable in a wider variety of situations. An interesting approach would be to use failures and repairs to form different levels of abstraction at which to represent plans.

(7) Detection and refinement of suboptimal and unnecessarily specific plans is another area for future work. Due to limited inferences or simplifications, it is possible that a learning system might not notice a fortuitous side effect of certain of its actions. As a result, it might use inappropriate or even redundant actions in a plan. Alternatively, the system might not notice that a plan would be applicable in a situation in which it is in fact applicable. Ideally, a planning system would have the capability to detect and remedy either of these situations.

(8) The system presently has little original problem-solving capability. It would be desirable that the system have some ability to plan for preconditions when finding a plan that is almost immediately applicable. The system should also have the ability to plan to prevent at least some observed failures (presently the system must wait until observing a plan repair).

(9) Another area for research would be to extend the knowledge representation to a more expressive system. For example, using intervals instead of situations would greatly enhance the expressiveness of the system. Allowing specialized numerical reasoning or non-instantaneous actions would also make the system more powerful. However, each of these extensions would greatly increase the complexity of the inference and refinement processes.

I have argued that planning is a computationally demanding task. In non-trivial domains, the number of possibly useful operators and potentially relevant variables is enormous. Even in game domains, the combinatorics of brute-force computation are intractable.

Yet people routinely deal with this type of complexity in a wide variety of tasks. One way in which this is accomplished is by using simplifications in reasoning. But this requires the ability to detect and recover from incorrect usage of simplifications

This thesis has described an integrated approach to problem-solving and learning designed to deal with complexity. In this approach, the system uses simplifications in initially understanding observed operator sequences in order to make the process tractable. However, this introduces the possibility of learning flawed plans. In order to deal with this possibility, the system monitors the usage of learned plans. When a failure to achieve a goal occurs, the system begins a refinement process to prevent repetition of the failure. Because the basic domain theory used by the system is sound, any failure must be due to a faulty simplification. The system then queries the world to pinpoint the faulty assumption in the plan. The reasons for the error in the assumption are explained and generalized using Explanation-Based Learning techniques. The resulting failure representation allows the system to avoid repetition of the failure.

REFERENCES

[Bennett87] S. W. Bennett, "Approximation in Mathematical Domains ," *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy , August 1987.

[Chapman85] D. Chapman, "Planning for Conjunctive Goals," Technical Report 802, MIT AI Lab, Cambridge, MA, November 1985.

[Chien86] S. A. Chien, "A Failure-Driven Approach to Schema Refinement," Working Paper 81, Artificial Intelligence Research Group, Coordinated Science Laboratory, Urbana, IL, June 1986.

[Chien87a] S. A. Chien, "On Resource Constrained Inference," Working Paper 82, AI Research Group, Coordinated Science Laboratory, University of Illinois, Urbana, Il., February 1987.

[Chien87b] S. A. Chien, "Extending Explanation-Based Learning: Failure-Driven Schema Refinement," *Proceedings of the Third IEEE Conference on Artificial Intelligence Applications*, Orlando, Florida, February 1987. (Also appears as Technical Report UILU-ENG-87-2203, AI Research Group, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign.)

[DeJong81] G. F. DeJong, "Generalizations Based on Explanations," *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, B.C., Canada, August 1981, pp. 67-70.

[DeJong86] G. F. DeJong and R. J. Mooney, "Explanation-Based Learning: An Alternative View," *Machine Learning 1*, 2 (April 1986), pp. 145-176.

[Doyle80] J. Doyle, "A Model for Deliberation, Action, and Introspection," Technical Report 581, MIT AI Lab, Cambridge, MA, 1980.

[Doyle86] R. Doyle, "Constructing and Refining Causal Explanations from an Inconsistent Domain Theory," *Proceedings of the National Conference on Artificial Intelligence*, Philadelphia, PA, August 1986, pp. 538-544.

[Falkenhainer86] B. Falkenhainer, "Towards a General Purpose Belief Maintenance System," *Proceedings Second Workshop on Uncertainty and Probability in Artificial Intelligence*, Philadelphia, PA, August 1986.

[Fikes72] R. E. Fikes, P. E. Hart and N. J. Nilsson, "Learning and Executing Generalized Robot Plans," *Artificial Intelligence 3*, (1972), pp. 251-288.

[Gupta87] A. Gupta, "Explanation-based Failure Recovery," *Proceedings of the National Conference on Artificial Intelligence*, Seattle, WA, July 1987, pp. 606-610.

[Haddawy86] P. F. Haddawy, "A Variable Precision Logic Inference System Employing the Dempster-Shafer Uncertainty Calculus," Technical Report

UIUCDCS-F-86-959, M. S. Thesis, Department of Computer Science, University of Illinois, Urbana, IL, December 86.

[Hammond86] K. Hammond, "Case-based Planning: An integrated theory of planning, learning, and memory," Research report no. 488, Yale University, Dept. of Computer Science, New Haven, CT, 1986.

[Kedar-Cabelli87] S. T. Kedar-Cabelli, "Formulating Concepts According to Purpose," *Proceedings of the National Conference on Artificial Intelligence*, Seattle, WA, July 1987, pp. 477-481.

[Keller87] R. M. Keller, "Defining Operationality for Explanation-Based Learning," *Proceedings of the National Conference on Artificial Intelligence*, Seattle, WA, July 1987, pp. 482-487.

[Langley81] P. Langley, "Data-Driven Discovery of Physical Laws," *Cognitive Science 5*, 1 (1981), pp. 31-54.

[McCarthy69] J. McCarthy and P. J. Hayes, "Some Philosophical Problems from the Standpoint of Artificial Intelligence,"in *Machine Intelligence 4*, B. Meltzer and D. Michie (ed.), Edinburgh University Press, Edinburgh, Scotland, 1969.

[McCarthy86] J. McCarthy, "Applications of Circumscription to Formalizing Common-Sense Knowledge," *Artificial Intelligence 28*, (1986), pp. 89-116.

[Michalski83] R. S. Michalski and R. E. Stepp, "Learning from Observation: Conceptual Clustering,"in *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell and T. M. Mitchell (ed.), Tioga Publishing Company, Palo Alto, CA, 1983, pp. 331-363.

[Mitchell83] T. M. Mitchell, "Learning and Problem Solving," *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, West Germany, August 1983, pp. 1139-1151.

[Mitchell85] T. M. Mitchell, S. Mahadevan and L. I. Steinberg, "LEAP: A Learning Apprentice for VLSI Design," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, August 1985, pp. 573-580.

[Mitchell86] T. M. Mitchell, R. Keller and S. Kedar-Cabelli, "Explanation-Based Generalization: A Unifying View," *Machine Learning 1*, 1 (January 1986), pp. 47-80.

[Mooney85] R. J. Mooney and G. F. DeJong, "Learning Schemata for Natural Language Processing," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, August 1985, pp. 681-687.

[Mooney86] R. J. Mooney and S. W. Bennett, "A Domain Independent Explanation-Based Generalizer," *Proceedings of the National Conference on Artificial Intelligence*, Philadelphia, PA, August 1986, pp. 551-555.

[Mooney87] R. J. Mooney, "A General Mechanism for Explanation-Based Learning and Its Application to Narrative Understanding," PhD. Thesis in preparation, Department of Computer Science, University of Illinois, Urbana, IL, 1987.

[Newell63] A. Newell and H. A. Simon, "GPS, A Program That Simulates Human Thought,"in *Computers and Thought*, E. A. Feigenbaum and J. Feldman (ed.), McGraw-Hill, New York City, NY, 1963, pp. 279-293.

[O'Rorke87] P. V. O'Rorke, "Explanation-Based Learning via Constraint Posting and Propogation," PhD. Thesis, Department of Computer Science, University of Illinois, Urbana, Il., January 1987.

[Quinlan86] J. R. Quinlan, "Induction of Decision Trees," *Machine Learning 1*, 1 (1986), pp. 81-106.

[Rajamoney85] S. Rajamoney, G. F. DeJong and B. Faltings, "Towards a Model of Conceptual Knowledge Acquisition through Directed Experimentation," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, August 1985. (Also appears as Working Paper 68, AI Research Group, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign.)

[Rajamoney87] S. A. Rajamoney and G. F. DeJong, "The Classification, Detection and Handling of Imperfect Theory Problems," *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, August 1987.

[Reiter80] R. Reiter, "A Logic for Default Reasoning," *Artificial Intelligence 13*, 1-2 (April 1980), pp. 81-113.

[Rumelhart80] D. Rumelhart, "Schemata: The Building Blocks of Cognition,"in *Theoretical Issues in Reading Comprehension*, W. Brewer (ed.), Lawrence Erlbaum and Associates, Hillsdale, NJ, 1980, pp. 33-58.

[Sacerdoti77] E. Sacerdoti, *A Structure for Plans and Behavior*, American Elsevier, New York, 1977.

[Segre87a] A. M. Segre, "Explanation-Based Learning of Generalized Robot Assembly Tasks," Ph.D. Thesis, Department of Electrical and Computer Engineering, University of Illinois, Urbana, IL, January 1987. (Also appears as UILU-ENG-87-2208, AI Research Group, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign.)

[Segre87b] A. M. Segre, "On the Operationality/Generality Trade-off in Explanation-Based Learning," *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, August 1987.

[Shavlik87a] J. W. Shavlik and G. F. DeJong, "Analyzing Variable Cancellations to Generalize Symbolic Mathematical Calculations," *Proceedings of the Third IEEE Conference on Artificial Intelligence Applications*, Orlando, FL, February 1987.

[Shavlik87b] J. W. Shavlik, G. F. DeJong and B. H. Ross, "Acquiring Special Case Schemata in Explanation-Based Learning," *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, Seattle, WA, July 1987.

[Silver86] B. Silver, "Precondition Analysis: Learning Control Information,"in *Machine Learning: An Artificial Intelligence Approach, Vol. II*, R. S. Michalski, J. G. Carbonell and T. M. Mitchell (ed.), Morgan Kaufmann, Los Altos, CA, 1986, pp. 647-670.

[Winston86] P. H. Winston, "Learning by Augmenting Rules and Accumulating Censors,"in *Machine Learning: An Artificial Intelligence Approach, Vol. II*, T. M. Mitchell, J. G. Carbonell and R. S. Michalski (ed.), Morgan Kaufmann, Los Altos, CA, 1986, pp. 45-61.

APPENDIX A

SAMPLE SYSTEM PERFORMANCE

The feasibility of this approach has been demonstrated by the implementation of a prototype learning system that uses the simplification-based approach. The current implementation has been tested on a sequence of several examples. This sequence of examples demonstrates the systems ability to: 1) learn using simplifications; 2) correct flaws due to incorrect usage of simplifications; and 3) use previously learned representations of failures to understand preventive measures. Each example takes from one to four minutes to run, the majority of the time being spent garbage collecting.

In the first training example, the system is shown a plan to construct a widget. This is a simple assembly consisting of a gear mounted on a shaft that extends through a sheet (shown in Figure A-1). The system is shown a sequence of steps which include drilling holes in the gear and

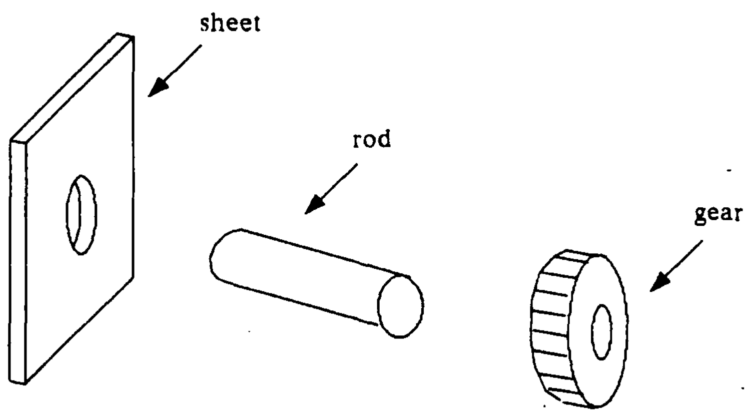


Figure A-1: Widget Assembly

sheet, and also heating and rolling a metal rod. The rod is then cooled and inserted into the gear for a friction fit, and the rod inserted through the sheet for a fit that allows the rod to spin. In this example, the gear is plastic and the other pieces are metal. The system learns a plan in which a number of defeasible inferences are used. The system learns this plan by constructing and generalizing an explanation using numerous defeasible and numerous completely sound rules of inference. However, in this explanation, the system uses a persistence assumption to verify that the shape of the gear does not change throughout the plan. Because of this assumption, the system does not understand that if the gear is plastic, the success of the plan depends upon the cooling step. The trace for this learning example is shown below:

```
Running: (RUN-EXAMPLE *W2*)
Initial State Consists of:
(AT R1 BENCH1)
(AT R2 BENCH1)
(AT G1 BENCH1)
(AT G2 BENCH1)
(AT P1 BENCH1)
(COMPOSITION R1 METAL)
(STATE R1 SOLID)
(STATE BASE SOLID)
(COMPOSITION G1 PLASTIC)
(SHAPE G1 GEAR)
(AT BASE BENCH1)
(SHAPE BASE SHEET)
Start state is state 0
operator 1 is (MOVE R1 OVEN)
operator 2 is (HEAT R1)
operator 3 is (MOVE R1 ROLLING-STATION)
operator 4 is (ROLL R1 10.0CM)
operator 5 is (COOL R1)
operator 6 is (MOVE R1 BENCH1)
operator 7 is (MOVE G1 DRILLING-STATION)
operator 8 is (DRILL G1 HOLE1 10.0CM)
operator 9 is (MOVE G1 BENCH1)
operator 10 is (MOVE BASE DRILLING-STATION)
operator 11 is (DRILL BASE HOLE2 10.1CM)
operator 12 is (MOVE BASE BENCH1)
operator 13 is (INSERT R1 G1 HOLE1)
operator 14 is (INSERT R1 BASE HOLE2)
```

```
Learning plan to achieve (WIDGET ?GEAR40645 ?PLANE37642 ?ROD41646 ?HOLE42647
?LOC38643 ?S139644).
Generalizing...
Packaging and Indexing...
Ordering Actions...
Marking assumption intervals...
Learned new plan WIDGET776
NIL
```

The system then uses this newly learned plan to solve several problems with all metal pieces. It successfully constructs several widgets. A successful problem-solving trace follows:

```
Running: (EXECUTE-EXAMPLE EX-SOLVE)
Start state is state 0
Initial State is:
(AT ROD101 BENCH3)
(AT GEAR212 BENCH2)
(AT BASE4 BENCH4)
(COMPOSITION ROD101 METAL)
(STATE ROD101 SOLID)
(SHAPE GEAR212 GEAR)
(STATE GEAR212 SOLID)
(SHAPE BASE4 SHEET)
(COMPOSITION GEAR212 METAL)
(STATE BASE4 SOLID)
Using plan WIDGET776 to achieve goal (WIDGET GEAR212 BASE4 ROD101 HOLE3 BENCH3
?S124752)
Operator 1 is (MOVE ROD101 OVEN)
Operator 2 is (HEAT ROD101)
Operator 3 is (MOVE ROD101 ROLLING-STATION)
Operator 4 is (ROLL ROD101 12.0CM)
Operator 5 is (MOVE ROD101 BENCH3)
Operator 6 is (MOVE GEAR212 DRILLING-STATION)
Operator 7 is (DRILL GEAR212 HOLE784 12.0CM)
Operator 8 is (MOVE GEAR212 BENCH3)
Operator 9 is (MOVE BASE4 DRILLING-STATION)
Operator 10 is (DRILL BASE4 HOLE785 12.1CM)
Operator 11 is (MOVE BASE4 BENCH3)
Operator 12 is (INSERT ROD101 GEAR212 HOLE784)
Operator 13 is (INSERT ROD101 BASE4 HOLE785)
Plan WIDGET776 completed.
Verifying achievement of functionality.
Goal achievement successfully verified.
NIL
```

However, the system is then requested to construct a widget with a plastic gear. The system attempts to apply the faulty plan. After the plan is executed, the system determines that the plan has failed. The system explains the failure and deduces that the plan will not work in cases where the gear is made of plastic because it will be deformed by being brought into contact with the heated rod. The system subsequently modifies the plan to reflect this limitation. The trace for this example follows:

```
Running: (EXECUTE-EXAMPLE FAIL-EX)
Start state is state 0
Initial State is:
(AT R1 BENCH2)
(AT G1 BENCH3)
(AT BASE BENCH4)
(COMPOSITION R1 METAL)
(STATE R1 SOLID)
(SHAPE G1 GEAR)
(SHAPE BASE SHEET)
(COMPOSITION G1 PLASTIC)
(STATE G1 SOLID)
(STATE BASE SOLID)
Using plan WIDGET776 to achieve goal (WIDGET G1 BASE R1 HOLE3 BENCH3 ?S124752)
Operator 1 is (MOVE R1 OVEN)
Operator 2 is (HEAT R1)
Operator 3 is (MOVE R1 ROLLING-STATION)
Operator 4 is (ROLL R1 15.0CM)
Operator 5 is (MOVE R1 BENCH3)
Operator 6 is (MOVE G1 DRILLING-STATION)
Operator 7 is (DRILL G1 HOLE787 15.0CM)
Operator 8 is (MOVE G1 BENCH3)
Operator 9 is (MOVE BASE DRILLING-STATION)
Operator 10 is (DRILL BASE HOLE788 15.1CM)
Operator 11 is (MOVE BASE BENCH3)
Operator 12 is (INSERT R1 G1 HOLE787)
Operator 13 is (INSERT R1 BASE HOLE788)
Plan WIDGET776 completed.
Verifying achievement of functionality.
Goal does not verify.  Backtracing supports.
Verifying (AT BASE BENCH3) in situation 13
(AT BASE BENCH3) in situation 13 Verified in real world
Verifying (SHAPE G1 GEAR) in situation 13
(SHAPE G1 DEFORMED) is not expected value, will investigate.
Verifying (SHAPE BASE SHEET) in situation 13
(SHAPE BASE SHEET) in situation 13 Verified in real world
Verifying (THROUGH R1 BASE HOLE788) in situation 13
(THROUGH R1 BASE HOLE788) in situation 13 Verified in real world
Verifying (DIAMETER R1 15.0CM) in situation 13
(DIAMETER R1 15.0CM) in situation 13 Verified in real world
Verifying (HOLE HOLE788 BASE 15.1CM) in situation 13
(HOLE HOLE788 BASE 15.1CM) in situation 13 Verified in real world
Verifying (THROUGH R1 G1 HOLE787) in situation 13
(THROUGH R1 G1 HOLE787) in situation 13 Verified in real world
Verifying (DIAMETER R1 15.0CM) in situation 13
(DIAMETER R1 15.0CM) in situation 13 Verified in real world
Verifying (HOLE HOLE787 G1 15.0CM) in situation 13
(HOLE HOLE787 G1 15.0CM) in situation 13 Verified in real world
Attempting to explain (SHAPE G1 DEFORMED) in situation 13
Added censor SHAPE31232.
NIL
```

Now the system does not have any plans for constructing widgets with plastic gears. When confronted with a problem-solving situation of this type, it cannot achieve the goal. A trace of this scenario follows:

```
Running: (EXECUTE-EXAMPLE NO-SOLVE)
Start state is state 0
Initial State is:
(AT ROD1 BENCH2)
(AT GEAR1 BENCH2)
(AT BASE BENCH4)
```

```
(COMPOSITION ROD1 METAL)
(STATE ROD1 SOLID)
(SHAPE GEAR1 GEAR)
(STATE GEAR1 SOLID)
(SHAPE BASE SHEET)
(COMPOSITION GEAR1 PLASTIC)
(STATE BASE SOLID)
Could not find a plan to achieve (WIDGET GEAR1 BASE ROD1 HOLE3 BENCH3
?GOAL-TIME1106).
NIL
```

Next, the system is shown the original learning example with a plastic gear. Because the plan includes a cooling step, it succeeds. In this case, the system understands that the actions observed are those in its representation of the original plan, with the addition of a cooling step. The system also understands that the plan succeeded, in a case where the melted-gear failure applied. The system then attempts to explain why the melted-gear failure did not occur. It explains that the failure was blocked by the cooling step, and hence understands the purpose of the cooling step. A new plan is now learned with an appropriately constrained cooling step. The trace for this example follows.

```
Running: (RUN-EXAMPLE *W3*)
Initial State Consists of:
(AT R21 BENCH2)
(AT R2 BENCH1)
(AT G312 BENCH3)
(AT G2 BENCH1)
(AT P1 BENCH1)
(COMPOSITION R21 METAL)
(STATE R21 SOLID)
(STATE BASE16 SOLID)
(COMPOSITION G312 PLASTIC)
(SHAPE G312 GEAR)
(AT BASE16 BENCH1)
(SHAPE BASE16 SHEET)
Start state is state 0
operator 1 is (MOVE R21 OVEN)
operator 2 is (HEAT R21)
operator 3 is (MOVE R21 ROLLING-STATION)
operator 4 is (ROLL R21 10.0CM)
operator 5 is (COOL R21)
operator 6 is (MOVE R21 BENCH1)
operator 7 is (MOVE G312 DRILLING-STATION)
operator 8 is (DRILL G312 HOLE1 10.0CM)
operator 9 is (MOVE G312 BENCH1)
operator 10 is (MOVE BASE16 DRILLING-STATION)
operator 11 is (DRILL BASE16 HOLE2 10.1CM)
operator 12 is (MOVE BASE16 BENCH1)
operator 13 is (INSERT R21 G312 HOLE1)
operator 14 is (INSERT R21 BASE16 HOLE2)
Learning plan to achieve (WIDGET ?GEAR401320 ?PLANE371317 ?ROD411321 ?HOLE421322
?LOC381318 ?S1391319).
Generalizing...
Packaging and Indexing...
Ordering Actions...
Comparing to previously learned plans.
Checking previously learned censors.
Investigating fix to failure...
Detected prevention...
Marking assumption intervals...
Learned new plan WIDGET1451
NIL
```

With the system's current problem-solving knowledge, the system will apply the original plan without the cooling step, in cases where the gear is not plastic. This is because although both plans are applicable, the plan without the cooling step is easier to execute, having one less step. A trace of a widget with a metal gear being constructed follows.

```
Running: (EXECUTE-EXAMPLE EX-EASY)
Start state is state 0
Initial State is:
(AT ROD101 BENCH3)
(AT GEAR12 BENCH2)
(AT BASE BENCH4)
```

```
(COMPOSITION ROD101 METAL)
(STATE ROD101 SOLID)
(SHAPE GEAR12 GEAR)
(STATE GEAR12 SOLID)
(SHAPE BASE SHEET)
(COMPOSITION GEAR12 METAL)
(STATE BASE SOLID)
Using plan WIDGET776 to achieve goal (WIDGET GEAR12 BASE ROD101 HOLE3 BENCH3
?S124752)
Operator 1 is (MOVE ROD101 OVEN)
Operator 2 is (HEAT ROD101)
Operator 3 is (MOVE ROD101 ROLLING-STATION)
Operator 4 is (ROLL ROD101 12.0CM)
Operator 5 is (MOVE ROD101 BENCH3)
Operator 6 is (MOVE GEAR12 DRILLING-STATION)
Operator 7 is (DRILL GEAR12 HOLE1459 12.0CM)
Operator 8 is (MOVE GEAR12 BENCH3)
Operator 9 is (MOVE BASE DRILLING-STATION)
Operator 10 is (DRILL BASE HOLE1460 12.1CM)
Operator 11 is (MOVE BASE BENCH3)
Operator 12 is (INSERT ROD101 GEAR12 HOLE1459)
Operator 13 is (INSERT ROD101 BASE HOLE1460)
Plan WIDGET776 completed.
Verifying achievement of functionality.
Goal achievement successfully verified.
NIL
```

However, in cases where the gear is plastic, only the cooling plan will be applicable, consequently the system will use that plan, as shown in the following trace.

```
Running: (EXECUTE-EXAMPLE EX-HARD)
Start state is state 0
Initial State is:
(AT ROD10 BENCH2)
(AT GEAR1 BENCH2)
(AT BASE BENCH4)
(COMPOSITION ROD10 METAL)
(STATE ROD10 SOLID)
```

```
(SHAPE GEAR1 GEAR)
(SHAPE BASE SHEET)
(COMPOSITION GEAR1 PLASTIC)
(STATE GEAR1 SOLID)
(STATE BASE SOLID)
Using plan WIDGET1451 to achieve goal (WIDGET GEAR1 BASE ROD10 HOLE3 BENCH3
?S1241427)
Operator 1 is (MOVE ROD10 OVEN)
Operator 2 is (HEAT ROD10)
Operator 3 is (MOVE ROD10 ROLLING-STATION)
Operator 4 is (ROLL ROD10 12.0CM)
Operator 5 is (COOL ROD10)
Operator 6 's (MOVE ROD10 BENCH3)
Operator 7 is (MOVE GEAR1 DRILLING-STATION)
Operator 8 is (DRILL GEAR1 HOLE1462 12.0CM)
Operator 9 is (MOVE GEAR1 BENCH3)
Operator 10 is (MOVE BASE DRILLING-STATION)
Operator 11 is (DRILL BASE HOLE1463 12.1CM)
Operator 12 is (MOVE BASE BENCH3)
Operator 13 is (INSERT ROD10 GEAR1 HOLE1462)
Operator 14 is (INSERT ROD10 BASE HOLE1463)
Plan WIDGET1451 completed.
Verifying achievement of functionality.
Goal achievement successfully verified.
NIL
T
>
```